Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматики
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка захищеного сховища даних із використанням технологій блокчейн»

(тема роботи)

Виконав: студент 2-го курсу гр. 1ІСТ-22м
(шифр групи)
спеціальності 126 – Інформаційні системи та технології
(шифр та назва спеціальності)
Степан СІДАК
(ім'я ПРІЗВИЩЕ студента)
Керівник: к.т.н., доц. каф. АІТ
Ярослав КУЛИК
(науковий ступінь, вчене звання / посада, ім'я ПРІЗВИЩЕ керівника)
«4» грудня 2023 р.
Опонент: к.т.н., доц. каф. Загальної фізики
Богдан КНИШ
(науковий ступінь, вчене звання / посада, ім'я ПРІЗВИЩЕ опонента)
«8» грудня 2023 р.

Допущено до захисту

Завідувач кафедри АІТ

Д.т.н., проф. Олег БІСІКАЛО

«11» грудня 2023 р.

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти II-ий (магістерський)
Галузь знань – 12 – Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри АПТ
д.т.н., проф. Олег БІСІКАЛО.
«20» вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Степану СІДАКУ

Тема роботи: Розробка захищеного сховища даних із використанням технологій блокчейн
Керівник роботи: к.т.н., доцент. каф. АПТ Ярослав КУЛИК
Затверджені наказом ВНТУ від «18» вересня 2023 року №247.
Строк подання студентом роботи до «05» грудня 2023 року.
Вихідні дані до роботи: кросплатформеність клієнтської та серверної частини; авторизація користувачів; використання шифрування при роботі з даними.
Зміст текстової частини: вступ; аналіз предметної області та огляд існуючих сховищ даних; опис системи та методи і моделі збереження даних; аналіз програмної реалізації захищеного сховища даних; програмна реалізація захищеного сховища даних; економічна частина; висновки; список використаних джерел.
Перелік ілюстративного (або графічного) матеріалу: uml діаграма класів; діаграма використання; схема архітектури додатку; блок схема алгоритму шифрування; вигляд основних сторінок додатку.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Ім'я, ПРІЗВИЩЕ та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ярослав КУЛИК, к.т.н., доцент каф. АІТ	20.09.23	04.12.23
5	Професор кафедри ЕПВМ, Професор, к.е.н Володимир КОЗЛОВСЬКИЙ	30.09.23	06.12.23

7. Дата видачі завдання « ____»_____2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	20.09.23 - 30.09.23	вик.
2	Вибір оптимальних моделей для збереження даних	01.10.23 - 10.10.23	вик.
3	Вибір мови програмування та середовища розробки	11.10.23 - 13.10.23	вик.
4	Розробка алгоритму шифрування даних	14.10.23 - 23.10.23	вик.
5	Програмна реалізація	26.10.23 - 01.11.23	вик.
6	Тестування розробленої системи	02.11.23 - 08.11.23	вик.
7	Економічна частина	26.10.23 - 06.12.23	вик.
8	Оформлення матеріалів до захисту МКР	13.11.23 - 03.12.23	вик.
9	Попередній захист роботи	12.12.23	вик.
10	Остаточний захист роботи	19.12.23	вик.

Студент _____ Степан СІДАК
(підпис)

Керівник роботи _____ Ярослав КУЛИК
(підпис)

АНОТАЦІЯ

УДК 004.75

Сідак С. В. Розробка захищеного сховища даних із використанням технологій блокчейн. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології. Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 108 с.
На укр. мові. Бібліогр.: 32 посилань; рис.: 55 табл.: 3.

У даному дослідженні було розроблено захищене сховище для зберігання даних із використанням шифрування – для забезпечення захисту вмісту, а також технологію блокчейн – для мінімізації можливості компрометації даних. Було запропоновано підхід до збереження файлів, який поєднує шифрування та використання блокчейну. На основі підходу, побудовано алгоритм та реалізовано додаток для роботи із даними, який використовує розроблений алгоритм роботи із даними.

Ключові слова: безпека даних, шифрування, блокчейн, розробка, сховище даних.

ANNOTATION

Sidak S. V. Development of a secure data storage using blockchain technologies. Master's thesis on specialty 126 - Information systems and technologies. Educational and professional program - Information technologies of data and image analysis. Vinnytsia: VNTU, 2023. 108 p.

In Ukrainian speech Bibliography: 32 links; fig.: 55 tab.: 3.

This study developed a secure data storage facility using encryption to ensure content protection and blockchain technology to minimize the possibility of data compromise. An approach to file storage that combines encryption and the use of blockchain has been proposed. Based on the approach, an algorithm was built and an application for working with data was implemented, which uses the developed algorithm for working with data.

Keywords: data security, encryption, blockchain, development, data storage.

ЗМІСТ

ВСТУП

З розвитком сучасних технологій та збільшенням обсягів інформації, яка використовується та зберігається користувачами, ефективне та безпечне зберігання даних стає надзвичайно важливим завданням для різних сфер діяльності. Необхідність забезпечення безпеки для персональних та загальних даних створює потребу в надійних засобах для зберігання.

Одним із перспективних напрямків розробки є використання технології блокчейн[1] для створення сховища даних[7]. Розробка сховища даних із використанням технологій блокчейн, забезпечує високу надійність, безпеку та швидкість доступу до інформації.

Використання технології блокчейн дозволяє забезпечити розподілене зберігання даних, що зменшує ризик втрати чи порушення інформації. Крім того, технологія блокчейн надає механізми цілісності даних, забезпечуючи високий рівень безпеки. У даній роботі будуть розглянуті основні аспекти розробки захищеного сховища даних із використанням технологій блокчейн, зокрема вибір алгоритмів, структур даних та механізмів контролю доступу.

Також буде розглянуто питання безпеки, надійності та можливого розвитку захищеного сховища даних.

Результатом цієї роботи буде розроблене захищене сховище даних із використанням технологій блокчейн, яке відповідатиме вимогам надійності, безпеки та ефективності.

Дана розробка має потенціал знайти практичне застосування в різних галузях, де зберігання даних мають вирішальне значення. Описана у роботі розробка захищеного сховища даних із використанням технологій блокчейн сприятиме поліпшенню якості зберігання даних та забезпечить їх безпеку.

Актуальність теми. Актуальність теми "Розробка сховища даних із використанням технологій блокчейн" полягає в тому, що в наші дні люди, організації та підприємства стикаються з великим обсягом даних, які потребують надійного та безпечного зберігання. Традиційні централізовані бази даних не завжди можуть

забезпечити необхідні рівні безпеки та прозорість, а також часто нехтують необхідними засобами безпеки, на користь зручності та швидкодії, що у перспективі виливається у втрату гігабайтів інформації. Звісно, це може бути невинне листування, проте, може бути і витік конфіденційної інформації, злив якої тягне за собою колосальні збитки. втрати даних.

Технологія блокчейн, в основі якої лежить розподілена мережа, відкриває нові перспективи для забезпечення безпеки, цілісності та незмінності даних. Вона дозволяє створювати сховища даних, в яких інформація захищена шифруванням та розподілена між багатьма вузлами мережі, що робить її важкодоступною для несанкціонованого доступу та модифікації.

Застосування технології блокчейн у сховищах даних може мати широкий спектр переваг, таких як:

- Забезпечення безпеки даних: Інформація в сховищі блокчейн захищена криптографічними методами, а дані розподіляються та реплікуються між вузлами мережі, що робить їх важкодоступними для несанкціонованого доступу.
- Незмінність даних: Записи в блокчейні є незмінними, що дозволяє перевіряти автентичність та цілісність інформації.
- Децентралізованість[2]: Блокчейн використовує розподілену мережу, що дозволяє уникнути централізованих точок вразливості та забезпечує високу доступність системи.

Розробка сховища даних із використанням технологій блокчейн є актуальним завданням, яке може сприяти покращенню безпеки та ефективності зберігання великих обсягів даних у різних галузях, включаючи фінанси, логістику, медицину та інші.

Окрім цього, в еру діджиталізації, важливим є забезпечення захисту для персональної інформації та файлів, таких як документи, фото та відео.

Мета та задача дослідження. Метою дослідження є зменшення ймовірності компрометації файлів та втрати приватних даних.

Задачі:

- Аналіз існуючих рішень для зберігання даних та можливості використання технології блокчейн.

- Розробка алгоритму захищеного зберігання даних із використанням шифрування та технологій блокчейн.

- Реалізація прототипу сховища даних з використанням блокчейн-технологій.

- Тестування та оцінка ефективності розробленого сховища даних.

Об'єкт дослідження. Об'єктом дослідження є технології блокчейн та їх застосування для розробки сховища даних.

Предмет дослідження. Предметом дослідження є процес розробки та впровадження сховища даних із використанням технологій блокчейн з метою забезпечення безпеки, незмінності та ефективного управління даними.

Методи дослідження. У процесі дослідження будуть використовуватись наступні методи:

- Аналіз літератури та наукових джерел для вивчення технологій блокчейн та їх застосування.

- Аналіз існуючих рішень та протоколів для зберігання даних на технології блокчейн.

- Розробка архітектури та дизайну сховища даних.

- Розробка програмного забезпечення для реалізації сховища даних.

- Тестування, валідація та оцінка ефективності розробленого сховища даних.

Науково-технічний результат. Науково-технічний результат даної роботи полягатиме у розробці сховища даних із відкритим вихідним кодом та із використанням нового підходу до зберігання даних із використанням комбінованого шифрування та технологій блокчейн, який забезпечує високий рівень безпеки, незмінності та прозорості інформації.

Практична цінність роботи. В результаті виконання роботи буде отримано web API, swagger документацію, веб клієнт, схему бази даних та алгоритм шифрування файлів із використанням каскадного шифрування та технології блокчейн.

Апробація результатів дослідження: основні результати виконання магістерської кваліфікаційної роботи було подано в матеріали науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, ВНТУ, 2023 р. - https://conferences.vntu.edu.ua/index.php/mn/mn2024).

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ СИСТЕМ СХОВИЩ ДАНИХ

1.1 Аналіз предметної області та особливості сховищ для зберігання даних

Стрімкий технологічний розвиток призвів до переходу багатьох систем, а в результаті даних, якими оперують ці системи у цифровий формат. Це Різноманітні таблиці із даними, текстові дані, фото та відео. Процес контролю доступу до цих даних потребує особливої уваги не тільки у контексті підприємства, яке може втратити важливі дані, які є їх інтелектуальною власністю, державних установ, які працюють із конфіденційними даними, а також простих людей, які в епоху інтернет втрачають своє право на конфіденційність[8] та приватне життя. Статистику про витоки даних відповідно до кількості втрачених записів та кількості постраждалих користувачів по різних роках проілюстровано на рисунку 1.1.

Characteristic	Number of impacted users and breached records
Cam4 Data Breach (Mar 2020)	10.88bn records
Yahoo Data Breach (2017)*	3bn acounts
Aadhaar Data Breach (Mar 2018)	1.1bn people
Alibaba Data Breach (Jul 2022)	1.1bn users
First American Financial Corporation Data Breach (May 2019)	885m users
Verifications.io Data Breach (Feb 2019)	763m users
LinkedIn Data Breach (Jun 2021)	700m users
Facebook Data Breach (Apr 2019)	533m users
Yahoo Data Breach (2014)	500m accounts
Satwood (Marriott) Data Breach (Nov 2018)	500m guests
Adult Freind Finder Data Breach (Oct 2016)	412.2m accounts
MySpace Data Breach (Jun 2013)	360m accounts
Exactis Data Breach (Jun 2018)	340m people

Рисунок 1.1 – Статистика про витоки даних

Використання сховища даних, яке дозволить забезпечити відповідний рівень безпеки для файлів захистить кінцевого користувача даної системи від проблем із приватністю, втрати конфіденційних даних, забезпечить збереження особистих таємниць.

Сучасні системи зберігання даних є складними інфраструктурами, які обробляють величезні обсяги інформації. Вони є життєво важливими для організацій та компаній, оскільки зберігають важливі дані, такі як фінансові записи, особисту інформацію, інтелектуальну власність та інші.

Однак, сучасні системи зберігання даних також піддаються численним ризикам і загрозам. Ці ризики можуть призвести до витоку даних, втрати даних або порушення доступу до даних.

Основні ризики та загрози, пов'язані з сучасними системами зберігання даних, включають:

Фізичні загрози містять у собі пожежі, повені, землетруси та інші природні явища, а також людський фактор і помилки в пристроях для зберігання даних.

Кібернетичні загрози включають атаки хакерів, шкідливі програми та віруси.

Організаційні загрози включають невідповідні політики та процедури роботи із даними, недостатню підготовку персоналу та відсутність контролю доступу до інформації.

Усі вищеперераховані загрози можуть призвести до витоку даних, втрати даних або порушення доступу до даних.

Для зниження ризиків і загроз, пов'язаних з сучасними системами зберігання даних, організації можуть вжити заходи для забезпечення належного рівня захисту інформації.

Наявність конкретного і відпрацьованого плану реагування на надзвичайні ситуації, впровадження комплексних заходів безпеки дозволить захистити системи зберігання даних від цих загроз.

Окрім цього, для зниження ризиків і загроз, пов'язаних з сучасними системами зберігання даних, вкрай важливо використовувати надійні систем зберігання даних, які забезпечать надійне зберігання та контроль доступу до даних і навіть у разі втрати даних – не дозволять зловмисникам отримати доступ до початкових даних. Шифрування даних може допомогти захистити дані від несанкціонованого доступу.

1.2 Актуальні проблеми конфіденційності та доступу до інформації

Конфіденційність та доступ до інформації є двома важливими принципами, які регулюють обробку даних. Конфіденційність передбачає захист даних від несанкціонованого доступу, їх використання або розкриття. Доступ до інформації передбачає можливість людей отримувати та зберігати інформацію.

У сучасному світі, коли обсяги даних постійно зростають, а технології роблять доступ до інформації все простішим, проблеми конфіденційності та доступу до інформації стають все більш актуальними.

Однією з найактуальніших проблем конфіденційності є зростання кіберзлочинності. Статистику по втраті даних за період 2020 - 2023 років зображено на рисунку 1.2.

Можемо помітити істотне зростання на кінці 2020 – початку 2021 року. Проте, величина втрати даних все одно залишається на високому рівні.

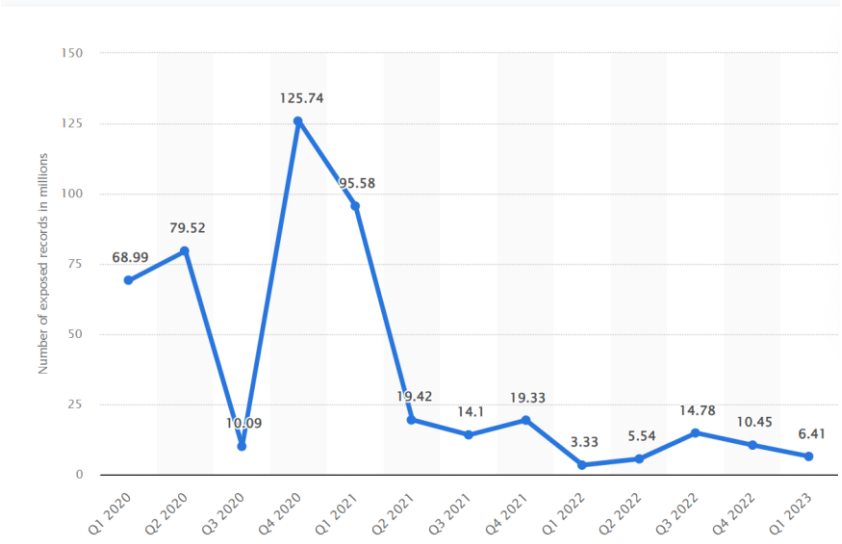

Рисунок 1.2 – Кількість втрачених даних за період 2020 – 2023 роки.

Кіберзлочинці використовують різні методи для отримання доступу до даних, зокрема:

Хакерські атаки - хакери використовують програмне забезпечення для злому систем безпеки та отримання доступу до даних.

Шкідливе програмне забезпечення - шкідливе програмне забезпечення може бути використане для крадіжки даних або контролю комп'ютера жертви.

Фішинг є видом соціальної інженерії, який використовується для обману людей з метою отримання їхніх особистих даних.

Кіберзлочинність може призвести до витоку даних, що може мати серйозні наслідки для людей, чиї дані були скомпрометовані. Наприклад, витік даних може призвести до крадіжки особистої інформації, фінансових даних або інтелектуальної власності, які згодом можуть бути використані для дискредитації конкретної особи, фінансового вимагання, нанесення ментальної та психологічної шкоди у вигляді кібербулінгу.

Іншою проблемою конфіденційності є використання даних для цілей таргетингу реклами. Компанії збирають дані про людей, щоб показати їм рекламу, яка, як вони вважають, буде для них цікавою, проте не кожен хоче щоб його дані аналізувалися корпораціями, та вважають це порушенням їхньої конфіденційності.

Однією з проблем доступу до інформації є зростання цензури. Деякі уряди обмежують доступ до інформації, яка, на їхню думку, може бути шкідливою або загрозливою. Наприклад, у деяких країнах доступ до певного контенту може бути заблокований. Використання захищеного сховища даних із використанням технологій дозволяє підвищити рівень захисту персональних та корпоративних даних та забезпечує надійний захист, шифруючи дані, та не даючи зловмисникам отримати доступ до цих даних. Проблеми конфіденційності та доступу до інформації є складними та багатогранними. Не існує простих рішень, які б могли вирішити ці проблеми. Однак важливо усвідомлювати ці проблеми та вживати заходів для їх вирішення.

1.3 Огляд технології блокчейн

Блокчейн[1] - це децентралізована[9] технологія зберігання інформації, що використовується в криптовалютах та інших онлайн-додатках. Вона реалізується у вигляді ланцюжка блоків, кожен із яких містить набір транзакцій та підписів, а також

хеш від попереднього блоку. Це робить блокчейн незмінним та захищеним від внесення змін без згоди всіх учасників мережі.

Технологія блокчейн розв'язує проблему довіри між учасниками мережі, оскільки всі учасники мають доступ до повної історії транзакцій, що робить підробку або зміну даних неможливим. Це дозволяє створити систему, яка працює без потреби у третій стороні чи посереднику.

Структура та дизайн блокчейну можуть змінюватись в залежності від конкретного варіанту використання та вимог, але, як правило, вони включають:

Блоки[2] - це набір даних, що містить набір транзакцій та унікальний хеш. Блок включає посилання на хеш попереднього блоку, створюючи ланцюжок блоків або блокчейн. Цей ланцюжок зберігається одночасно в різних місцях і містить дані про всі транзакції. Щоб підробити транзакцію в одному блоці, треба змінити всі файли, що слідують за ним, але це зробити практично неможливо.

Блокчейн транзакції[25] - це записи про операції між двома чи більше сторонами. У блокчейні транзакції групуються у блоки та додаються в ланцюжок.

Вузли чи ноди - це комп'ютери, на яких працює програмне забезпечення, необхідне для підтримки мережі блокчейну. Вони видобувають нові блоки, підтверджують або скасовують транзакції, передають дані мережею та зберігають копії всього ланцюжка блоків. Як нода в блокчейні може виступати будь-яка людина, яка встановила необхідне програмне забезпечення блокчейну.

Блокчейн є специфічним типом технології розподіленого реєстру (DLT - Distributed Ledger Technology). DLT[2] діє як децентралізована база даних інформації про транзакції між різними сторонами. Операції заповнюють DLT у хронологічному порядку та зберігаються у вигляді серії блоків. Між блоками формується взаємопов'язаний ланцюжок, кожен з яких посилається на попередній блок, таким чином створюючи блокчейн.

У блокчейн-сховищі файли спочатку розбиваються на частини в процесі, який називається шардингом. Кожен шард копіюється, щоб запобігти втраті даних у разі виникнення помилки під час передачі. Файли також зашифровані за допомогою закритого ключа, який унеможливлює їх перегляд іншими вузлами в мережі.

Тиражовані сегменти розподіляються між децентралізованими вузлами по всьому світу. Взаємодії реєструються в журналі блокчейну, що дозволяє системі підтверджувати та синхронізувати транзакції між вузлами блокчейну.

Блокчейн-сховище призначене для збереження цих взаємодій назавжди, і дані ніколи не можуть бути змінені.

1.4 Використання блокчейну для збереження даних

Блокчейн - це розподілена база даних або книга, яка спільно використовується вузлами комп'ютерної мережі. Вони найбільш відомі своєю ключовою роллю в системах крипто валют для підтримки безпечного та децентралізованого запису транзакцій, але вони не обмежуються використанням крипто валюти. Блокчейни можна використовувати, щоб зробити дані в будь-якій галузі незмінними. Оскільки неможливо змінити блок, користувач або система може бути певним у валідності даних.

З моменту появи Bitcoin в 2009 році використання блокчейну зросло завдяки створенню різноманітних крипто валют, програм децентралізованого фінансування (DeFi), незамінних токенів (NFT) і смарт-контрактів[13].

Різні типи інформації можуть зберігатися в блокчейні, але найпоширеніше використання досі було в якості облікової книги для транзакцій. У випадку Bitcoin блокчейн використовується децентралізованим способом, щоб жодна людина або група не мали контролю над даними, всі користувачі колективно зберігають контроль. Децентралізовані блокчейни незмінні, а це означає, що введені дані незворотні. Для Bitcoin транзакції постійно реєструються та доступні для перегляду будь-кому.

Блокчейн дещо схожий на базу даних, у яку вводиться та зберігається інформація. Але ключова відмінність між традиційною базою даних та блокчейном полягає в тому, як дані структуровані та доступні.

Блокчейн складається з програм, які називаються сценаріями, які виконують завдання, які зазвичай виконуються в базі даних: введення інформації та отримання

доступу до неї, збереження та зберігання її десь. Блокчейн розподіляється, що означає, що кілька копій зберігаються на багатьох машинах, і всі вони повинні збігатися, щоб він був дійсним.

Блокчейн збирає інформацію про транзакції та вводить її в блок, як у клітинку в електронній таблиці, що містить інформацію. Після заповнення , інформація проходить через алгоритм шифрування, який створює хеш.

Потім хеш вводиться в наступний заголовок блоку та шифрується з іншою інформацією в блоці. Це створює серію блоків, які з'єднані разом.

Транзакції слідують за певним процесом, залежно від блокчейну, на якому вони відбуваються. Наприклад, у блокчейні Bitcoin, при ініціалізації транзакції за допомогою крипто валютного гаманця - програми, яка надає інтерфейс для блокчейну, відбувається послідовність подій: у Bitcoin транзакція надсилається до пулу пам'яті, де вона зберігається та ставиться в чергу, доки її не підбере майнер або валідатор. Після того, як його введено в блок і блок заповниться транзакціями, він закривається та шифрується за допомогою алгоритму шифрування.

Процес виконання транзакції зображено на рисунку 1.3

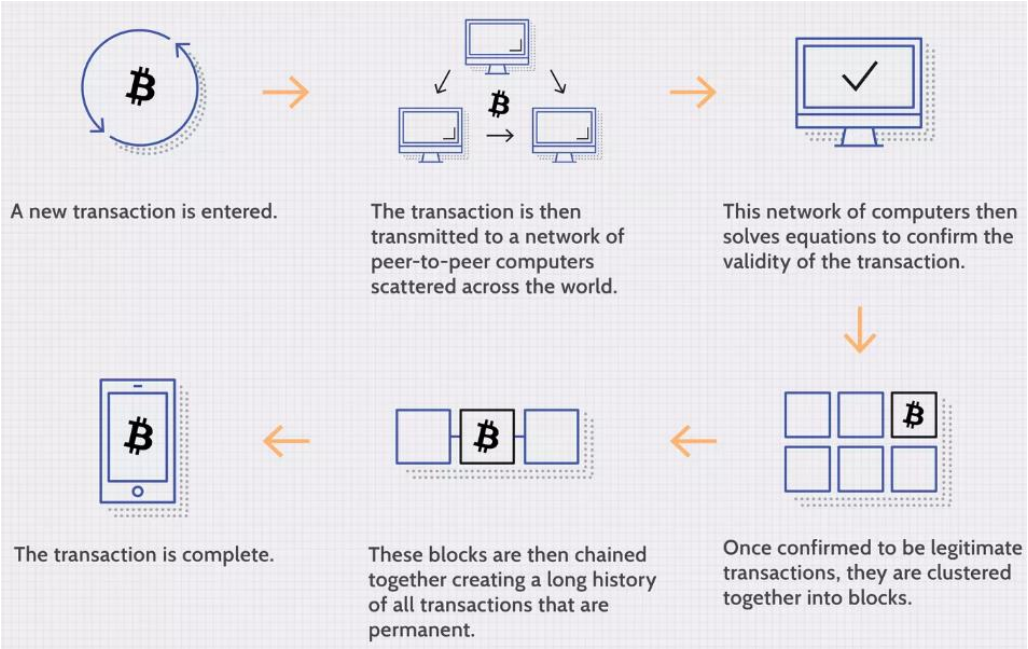

Рисунок 1.3 – Процес виконання транзакції

Блокчейн дозволяє розподіляти дані в базі даних між кількома вузлами мережі - комп'ютерами або пристроями, на яких працює програмне забезпечення для

блокчейну - у різних місцях. Це не тільки створює надмірність, але й підтримує точність даних. Наприклад, якщо хтось спробує змінити запис в одному екземплярі бази даних, інші вузли завадять цьому. Таким чином, жоден окремий вузол у мережі не може змінити інформацію, що зберігається в ньому. Завдяки цьому розповсюдженню - і зашифрованому доказу того, що робота була виконана - інформація та історія (як і транзакції в крипто валюті) незворотні. Таким записом може бути список транзакцій (наприклад, із крипто валютою), але блокчейн також може зберігати різноманітну іншу інформацію, залежно від реалізації. Це можуть бути ідентифікатори операцій, хеші, посилання тощо. На прикладі файлів – можна зберігати невеликі файли, на виборах – результати голосування, у державних установах - посвідчення, акти, накази та багато іншого.

1.5 Огляд існуючих сховищ даних

Сховища даних за їх інфраструктурним розміщенням можна поділити на мережеві та локальні. Відповідно для того, щоб скористатися мережевим сховищем даних, необхідно мати доступ до мережі інтернет, а офлайн сховище вимагає тільки локального встановлення на відповідному комп’ютері.

Azure Blob Storage - це рішення Microsoft для зберігання об’єктів у хмарі. Blob Storage оптимізовано для зберігання величезних обсягів неструктурованих даних. Неструктуровані дані – це дані, які не відповідають певній моделі даних або визначенню, наприклад текстові або двійкові дані.

Blob Storage пропонує три типи ресурсів:

Обліковий запис – створюється як основна одиниця для збереження даних, містить в собі контейнери, які можуть бути заточені під зберігання контейнерів із різними даними, проте рекомендовано мати окремі облікові записи для різних додатків.

Контейнер в обліковому записі – створюється для збереження певних даних, згрупованих за типом чи призначенням, не варто змішувати дані між собою, а краще групувати їх залежно від відповідальності того чи іншого контейнеру.

Blob – безпосередньо містить в собі всі файли.

На Рисунку 1.4 показано зв’язок між цими ресурсами.

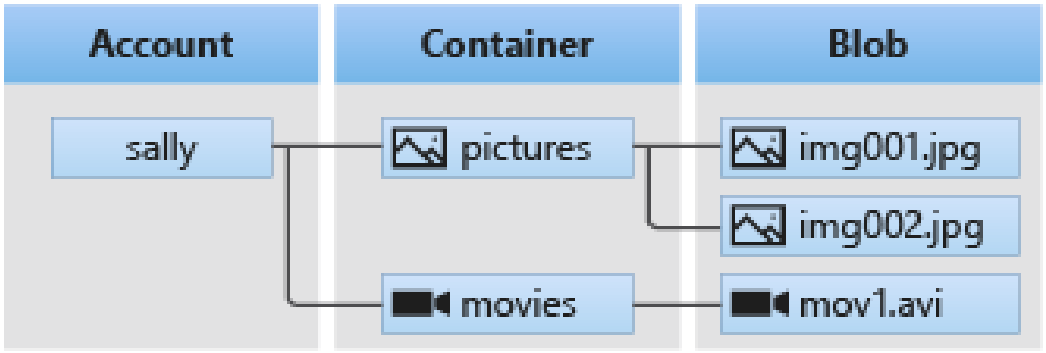

Рисунок 1.4 – Зв’язок між ресурсами Azure Blob

Azure Blob Storage підтримує три основних типи контейнерів[3], включаючи загальнодоступні контейнери (public), приватні контейнери (private) та контейнери для обмежених доступів (restricted). Це дозволяє точно налаштувати рівень доступу до сховища даних.

Клієнтські бібліотеки доступні для різних мов, зокрема:

- .NET[6];
- Java;
- Node.js;
- Python.

Сховище може масштабуватися в залежності від потреб проекту, але зберігається у хмарі, що дозволяє легко масштабувати параметри сховища.

Azure Blob Storage надає можливості шифрування даних для забезпечення конфіденційності та захисту інформації[17]. Дані шифруються як при зберігання, так і під час передачі між користувачем та сховищем. Є два основних типи шифрування:

Шифрування в спокійному часі (Encryption at Rest) - дані зберігаються в зашифрованому вигляді на серверах Azure Blob Storage.

Шифрування під час передачі (Encryption in Transit) - дані, які передаються між користувачем та Azure Blob Storage, також шифруються для захисту їх під час передачі по мережі.

Окрім цього, Azure Blob Storage[3] дозволяє налаштовувати рівні доступу до сховища даних і контролювати, хто і як може звертатися до них. Це допомагає забезпечити безпеку даних та зменшити ризики несанкціонованого доступу.

Інтеграція з іншими послугами Azure: Azure Blob Storage легко інтегрується з іншими послугами Azure, такими як Azure Functions, Azure Logic Apps та Azure Data Lake Storage, що робить його потужним інструментом для розробки хмарних додатків в інфраструктурній системі Azure.

Підсумовуючи, Azure Blob Storage - це потужний інструмент, який доцільно використорвувати у великих проектах, він спрямований на використання у купі з додатком, який працює із файлами, потребує постійного інтернет з'єднання та платної підписки. Основні недоліки –дорогий, потребує саме серверів Azure, не підходить для малих додатків та використання локально, на персональному комп’ютері

Google - це компанія, яка надає широкий спект різноманітних послуг. За збереження даних в екосистемі сервісів Google відповідає хмарний сервіс Google Drive. Google запустив Google Drive у 2012 році, і з тих пір він став одним із найбільш використовуваних хмарних сховищ[14]. Великі об'єми даних зберігаються на його серверах. Google диск є одним із найбільш універсальних варіантів хмарного сховища, для зберігання персональних файлів. Він пропонує інтеграцію з багатьма онлайн-програмами, має інтеграцію із сервісами Google Workspace. Такими як Google Docs, Sheets і Slides.

Вигляд інтерфейсу користувача у середовищі Google drive зображено на рисунку 1.5.

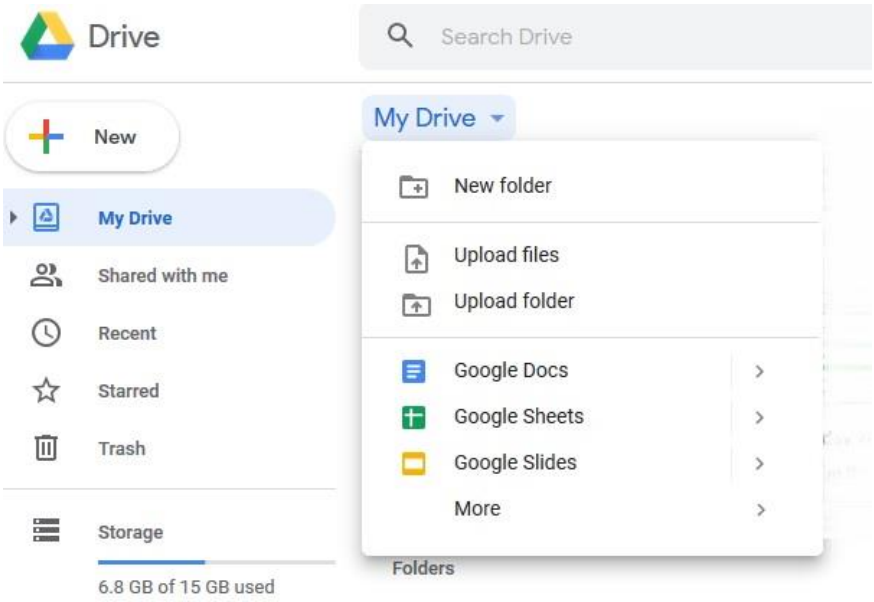

Рисунок 1.5 – Інтерфейс Google drive

Однак ставлення Google до конфіденційності завжди викликало у деяких людей здивування. Навіть незважаючи на те, що його безпека на найвищому рівні, користувацькі дані можуть бути доступні працівникам або алгоритмам Google.

Google Drive пропонує чудовий безкоштовний план із 15 ГБ вільного місця для зберігання, доступним для всіх користувачів. Крім того, він не обмежує жодні функції для безкоштовних користувачів - на відміну від більшості інших хмарних служб, що робить його одним із найкращих безкоштовних хмарних сховищ.

Amazon S3, пропонує гарну інфраструктуру, хороші швидкості, але нечітку структуру ціноутворення[15]. Amazon Simple Storage Service (або Amazon S3) пропонує підприємствам зберігати об’єкти, надаючи їм недороге онлайн-сховище з можливістю масштабування. AWS є популярним постачальником інфраструктури як послуги (IaaS) сьогодні. Він надає веб-хостинг, хмарні[4] обчислення та послуги розробки, об’єднані в один взаємопов’язаний пакет. Amazon S3 є важливою частиною послуг, оскільки він зберігає всі необхідні файли. Основна перевага S3 помітна, при використанні його разом із іншими веб-службами Amazon, з якими він бездоганно інтегрований.

Хоча в теорії, його безпека хороша, складність AWS призводить до порушень безпеки, а підтримка клієнтів спеціалістами AWS вимагає окремої платні.

Amazon S3 - це платформа хмарних обчислень, що означає, що вона забезпечує інфраструктуру, необхідну для бізнесу для впровадження онлайн-рішень. Це може означати багато різних речей. Це може бути спосіб просто зберігати дані, як альтернатива хмарному сховищу для бізнесу або спосіб резервного копіювання даних для архівних цілей, ніби це онлайн-служба резервного копіювання. Однак Amazon S3 є найбільш корисним як сховище для зберігання даних, необхідних для створення та розміщення вмісту, наприклад веб-сайтів і програм. Це особливо актуально, якщо S3 використовуєте інші служби AWS для створення або розміщення цього вмісту. На відміну від класичних хмарних служб зберігання файлів, Amazon S3 пропонує об’єктне сховище, тобто коли файл завантажується на його сервери, він перетворюється на об’єкт. Об’єкти схожі на файли, але вони можуть зберігати набагато детальніші метадані, що веде до кращої організації даних і точного глибокого пошуку. Політику доступу до ресурсів Amazon S3 зображено на рисунку 1.6.

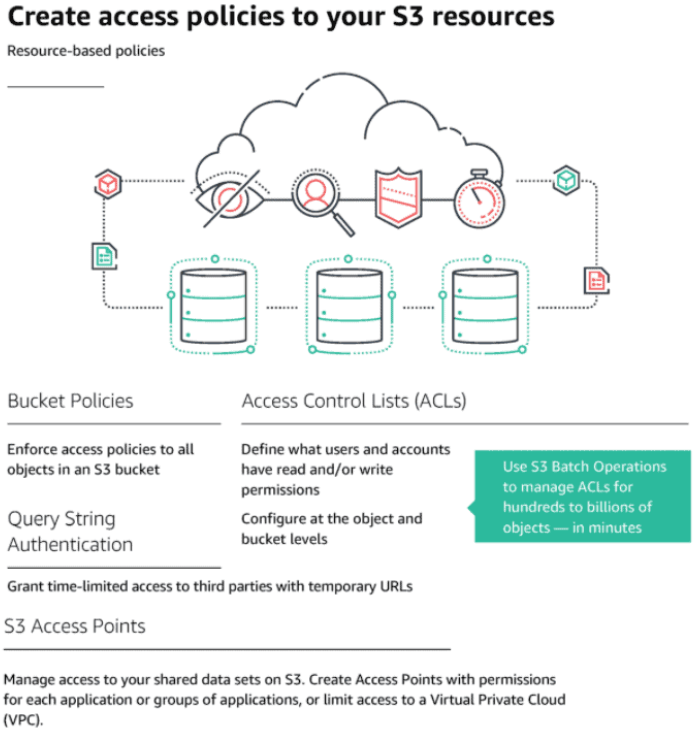

Рисунок 1.6 – Політика доступу до ресурсів Amazon S3

Amazon S3 стягує плату не лише за зберігання файлу, а й за його використання. У деяких випадках також варто перевірити наявність плати за використання API.

Чотири плани зберігання Amazon S3 називаються S3 Standard, S3 Intelligent – Tiering, S3 Standard – Infrequent Access і S3 One Zone – Infrequent Access. Реєстрація облікового запису S3 безкоштовна, є можливість зберігати до 5 ГБ безкоштовно протягом першого року.

Інтелектуальне багаторівневе ціноутворення є особливим із точки зору складності. Є чотири рівні зберігання – частий доступ, нечастий доступ, архів і глибокий архів - і стягує додаткову плату за моніторинг і автоматизацію.

Він має три рівні частого доступу відповідно до використання пам’яті, які відображають ціни стандартного плану. На щастя, решта його цінових рівнів мають фіксовану ціну.

Мережева інфраструктура Amazon S3 є важливою причиною того, що сервіс зберігає свої позиції лідера ринку IaaS. Загалом Amazon S3 має 25 центрів обробки даних по всьому світу, що лише на два сервери менше, ніж у мережі Google Cloud. Хоча й не така велика, як мережа Microsoft Azure, швидкість копіювання файлів до та з мережі S3 і мультирегіональне сховище є перевагою. Дані про розміщення центрів інфраструктури Amazon S3 зображено на рисунку 1.7.

Рисунок 1.7 – Розташування інфраструктури Amazon

Amazon робить багато кроків, щоб забезпечити безпеку даних S3, включаючи опцію встановлення стандартного шифрування для сегментів зберігання. Це приємне

доповнення, оскільки багато постачальників IaaS забезпечують лише захист під час передачі та покладають на плечі користувача шифрування на стороні сервера.

Протокол шифрування - стандартний AES, який використовується багатьма хмарними службами, встановлений на 256 біт.

Оскільки слабкі паролі є однією з найбільш часто використовуваних дірок у безпеці, Amazon також підтримує багатофакторну автентифікацію (MFA). Коли MFA увімкнено, під час входу потрібно буде ввести додатковий код безпеки. Цей код надходить на мобільний телефон у текстовому вигляді.

Незважаючи на те, що існує багато проблем з безпекою щодо хмарного рішення Amazon Cloud Drive, S3 виглядає надійним та безпечним.

Dropbox зробив хмарне сховище загальнодоступним. Він продовжує розвиватися з моменту свого запуску в 2008 році[16]. Однак компанія могла б покращити конфіденційність. Наприклад, відсутність шифрування з нульовим знанням може змусити деяких звернутись до альтернатив Dropbox.

Dropbox пропонує функції, які сподобаються як особистим, так і бізнес-користувачам. Для бізнесу інтегровані інструменти, такі як Google Docs і Microsoft Word.

Користувацький інтерфейс Dropbox зображено на рисунку 1.8

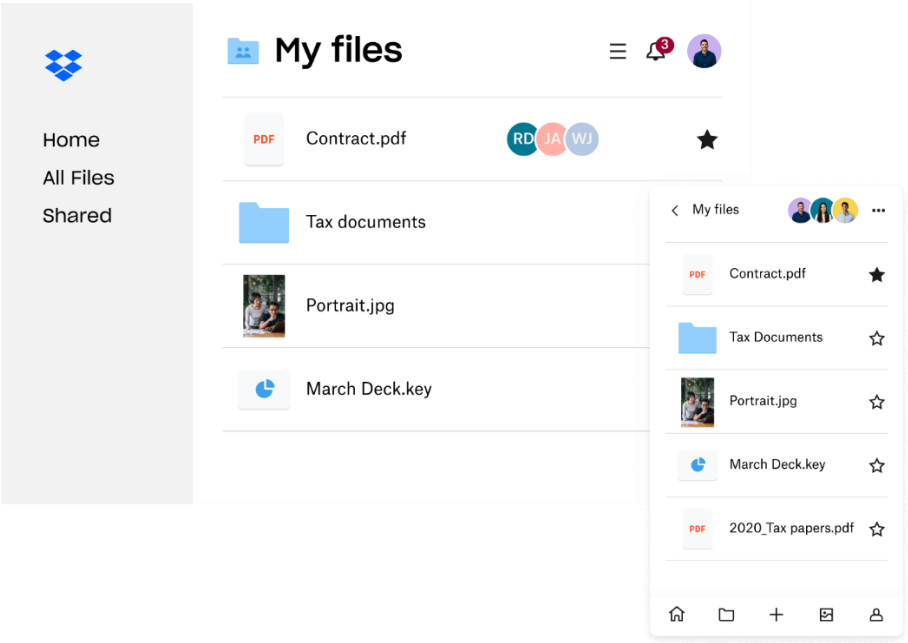

Рисунок 1.8 – Користувацький інтерфейс Dropbox

Сервіс Dropbox відстає від конкурентів, коли справа доходить до безпеки. Dropbox зберігає ключі шифрування на своїх серверах, що потенційно дозволяє отримати до них доступ стороннім особам.

Відсутність шифрування з нульовим знанням означає, що співробітники Dropbox, правоохоронні органи та хакери мають більше шансів отримати доступ до файлів.

Dropbox використовує галузевий стандарт 256-бітного шифрування AES для захисту даних у стані спокою та 128-бітного шифрування AES для даних під час передачі. Dropbox також реалізує двофакторну автентифікацію для додаткового рівня захисту.

У 2012 році в Dropbox стався масштабний витік даних, що призвело до витоку 68 мільйонів паролів користувачів Dropbox. Відтоді не було жодних ознак злому і захист Dropbox покращився.

У політиці конфіденційності Dropbox дуже чітко зазначено, що він може отримати доступ до даних, щоб переконатися, що його політику не було порушено. У ньому також зазначено, що він передаватиме дані довіреним третім сторонам, таким як Google, Amazon і Oracle.

Filecoin - це криптовалюта, яка спрямована на заохочення глобальної мережі власників комп’ютерів до надання послуг зберігання та обміну файлами.

Засновники проекту стверджують, що це може стати найшвидшим і найдешевшим способом зберігання даних в Інтернеті за умови участі достатньої кількості людей. Крім того, він не передбачає централізованого контролю, а це означає, що обмін файлами не підлягатиме цензурі з боку державних органів чи інших організацій.

Filecoin працює за допомогою майнерів, які виділяють ресурси для виконання обчислень, необхідних для роботи мережі. Майнери Filecoin отримують гроші за надання користувачам місця для зберігання. Користувачі Filecoin, у свою чергу, повинні платити майнерам за зберігання, пошук або розповсюдження цих даних.

Мережа Filecoin працює на основі іншого протоколу для децентралізованої обробки файлів під назвою Interplanetary File System (IPFS).

Filecoin - не єдиний протокол, який претендує на роль децентралізованої системи зберігання та обміну файлами на основі технологій криптовалюти.

Конкуруючі протоколи включають Storj і Siacoin. Команда Storj стверджує, що проект досяг ємності мережі понад 100 петабайт, тоді як Siacoin повідомляє про ємність 2 петабайти станом на 2020 рік.

Filecoin - це щось на зразок Dropbox, але на блокчейні. Користувачі, які хочуть зберігати будь-які дані в мережі Filecoin, повинні платити за це майнерам.

Ця система працює наступним чином:

Сума, яку вони заплатять, визначається відкритим ринком, де майнери змагаються один з одним, щоб запропонувати найнижчу ціну за зберігання. Команда Filecoin стверджує, що цей ринок буде «гіперконкурентним» і, отже, дешевшим, ніж централізоване зберігання даних, таке як Amazon Web Services або Microsoft Azure Blob.

У майнерів, у свою чергу, є стимул надавати сховище, оскільки вони можуть отримувати винагороду від мережі у вигляді токенів Filecoin. Чим більше пам’яті вони пропонують мережі, тим більша ймовірність того, що вони отримають винагороду. Але ця винагорода не дається задарма. Майнер повинен виконати кілька інтенсивних обчислювальних процесів (так звані докази), щоб довести мережі, що він дійсно зберігає дані, які він заявляє, і робить це надійно протягом певного періоду часу. Якщо майнер робить це надійно і пропонує достатньо місця для зберігання, він може створювати нові блоки в блокчейні Filecoin і отримувати мережеві винагороди та комісії за транзакції.

Блокчейни використовують механізми підтвердження концепції, щоб забезпечити узгодження нових транзакцій між усіма користувачами мережі. Наприклад, блокчейн біткойн покладається на підтвердження роботи, коли майнер повинен довести, що він виконав величезний обсяг обчислень, щоб мати право додавати нові транзакції в блокчейн і отримати щойно створені біткоїни.

Протокол Filecoin використовує два нових типи доказів, щоб підтвердити, що майнери дійсно зберігають дані, які вони заявляють. Доказ реплікації (Proof-of-Replication) доводить, що майнер дійсно зберіг ту кількість копій даних, про яку він

заявляє. Доказ простору та часу (Proof-of-Spacetime) доводить, що майнер зберігав дані протягом узгодженого періоду часу. Разом ці докази дозволяють користувачам довіряти майнерам у тому, що вони дійсно зберігають ті дані, які вони заявляють.

Детальне порівняння за основними характеристиками для розглянутих сервісів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння розглянутих сервісів за основними характеристиками

Характеристика	Filecoin	Azure Blob Storage	Google Drive	Dropbox	Amazon S3
Інтеграція з іншими сервісами	-	+	+	+	+
Ціна	+ (зазвичай менше)	- (зазвичай доступний безкоштовний обсяг)	- (вартість може бути вищою)	+ (різні опції тарифів)	+ (різні опції тарифів)
Масштабованість	+	+	+	+	+
Підтримка платформ	-	+	+	+	+
Швидкість передачі даних	+	+	+	+	+
Можливості роботи з версіями файлів	-	+	+	+	+
Наявність API	+	+	+	+	+
Безпека	+	+	+	+-	+

Продовження таблиці 1.1

Географічна розподіленість даних	+	+	+ (залежить від регіону)	+	+
Можливості роботи зі зберіганням	+	+	+	+	+
Шифрування	+	+	+	+	+
Використання блокчейн технологій	+	-	-	-	-
Можливості офлайн використання	+	+	+	+	+
Наявність коду у відкритому доступі	-	-	-	-	-
Мінуси	Залежність від блокчейн, менш розвинений екосистема	Вартість, менш розвинені можливості синхронізації та інтерфейс	Обмежений обсяг безкоштовного зберігання та ключі управління належать Google	Вартість підписки, обмежена інтеграція з іншими сервісами	Для новачка може здатися складним у використанні, ціна може збільшитися при збільшенні обсягу даних

1.6 Висновки

Аналіз предметної області показав, що існують нагальні проблеми в конфіденційності та безпеці зберігання файлів. Одним із методів боротьби із проблемами безпеки та конфіденційності як для конкретних осіб так і організацій є безпечне зберігання файлів. Впровадження шифрування при збереженні файлів може істотно знизити ризик неправомірного доступу до персональних та корпоративних даних, а також інтелектуальної власності. Використання технології блокчейн забезпечить контроль над даними, їх незмінність. Існуючі сховища для зберігання даних мають як недоліки так і переваги, проте серед них немає аналогу, який би поєднував безпеку даних за рахунок шифрування та технологію блокчейн. Блокчейн - це тип спільної бази даних, яка відрізняється від типової бази даних способом зберігання інформації; блокчейни зберігають дані в блоках, пов’язаних разом за допомогою криптографії.

2. ОПИС СИСТЕМИ ТА МЕТОДИ І МОДЕЛІ ЗБЕРЕЖЕННЯ ДАНИХ

2.1 Поняття дані, підходи до збереження даних

В обчислювальній техніці дані - це інформація, яку було переведено у форму, ефективну для переміщення або обробки. Відносно сучасних комп'ютерів і засобів передачі дані – це інформація, перетворена в двійкову цифрову форму. Необроблені дані – це термін, який використовується для опису даних у їх найпростішому цифровому форматі. Концепція даних у контексті обчислювальної техніки бере свій початок у роботах Клода Шеннона, американського математика, відомого як батько теорії інформації. Він започаткував двійкові цифрові концепції, засновані на застосуванні двозначної булевої логіки до електронних схем. Двійкові цифрові формати лежать в основі центральних процесорів, напівпровідникової пам'яті та дискових накопичувачів, а також багатьох периферійних пристроїв, поширених сьогодні в комп'ютерах. Ранній комп'ютерний вхід як для керування, так і для даних мав форму перфокарт, потім магнітної стрічки, жорсткого диска та твердотільного накопичувача. Комп'ютери представляють дані, включно з відео, зображеннями, звуками та текстом, у вигляді двійкових значень, використовуючи 1 і 0. Біт – це найменша одиниця даних, яка представляє лише одне значення, може мати значення 0 або 1. Байт складається з восьми бітів.

Файл - це набір байтів, який зберігається під певним іменем на пристрої зберігання. Файли можуть мати різні формати залежно від типу даних, які вони містять. Наприклад текстові файли (.txt), зображення (.jpg, .png), документи (.docx, .pdf) тощо. Формат файлу визначає, як дані структуровані. Файли організовані в ієрархічній структури у каталогах (папках). Ієрархічна файлова система дозволяє користувачам організовувати файли в деревоподібну структуру, полегшуючи навігацію та керування даними. При збереженні файлу на комп'ютері, дані записуються в певне місце на запам'ятовуючому пристрої за допомогою файлової системи, яка використовується. Операційна система відстежує розташування файлу, дозволяючи отримувати та змінювати дані пізніше. Поєднання байтів, пристроїв

зберігання даних, файлових систем і форматів файлів становить основу зберігання даних на персональних комп’ютерах.

Існують два основних типи сховищ для збереження даних: первинне та вторинне.

Первинне сховище, також відоме як оперативна пам’ять (RAM), є енергозалежним сховищем, яке тимчасово зберігає дані під час роботи комп’ютера. Оперативна пам’ять забезпечує швидкий доступ до даних, дозволяючи комп’ютеру ефективно обробляти та виконувати програми. Однак дані, що зберігаються в оперативній пам’яті, втрачаються, коли комп’ютер вимикається.

Вторинне сховище, також відоме як масове сховище, є енергонезалежним сховищем, яке постійно зберігає дані, навіть коли комп’ютер вимкнено. Звичайні вторинні пристрої зберігання включають жорсткі диски (HDD) і твердотільні накопичувачі (SSD). Жорсткі диски використовують обертові пластини для зберігання даних, тоді як твердотільні накопичувачі використовують чіпи флеш-пам’яті. SSD забезпечують більш високу швидкість доступу до даних порівняно з жорсткими дисками, але загалом вони дорожчі. Процес збереження даних у пристроях комп’ютеру проілюстровано на рисунку 2.1.

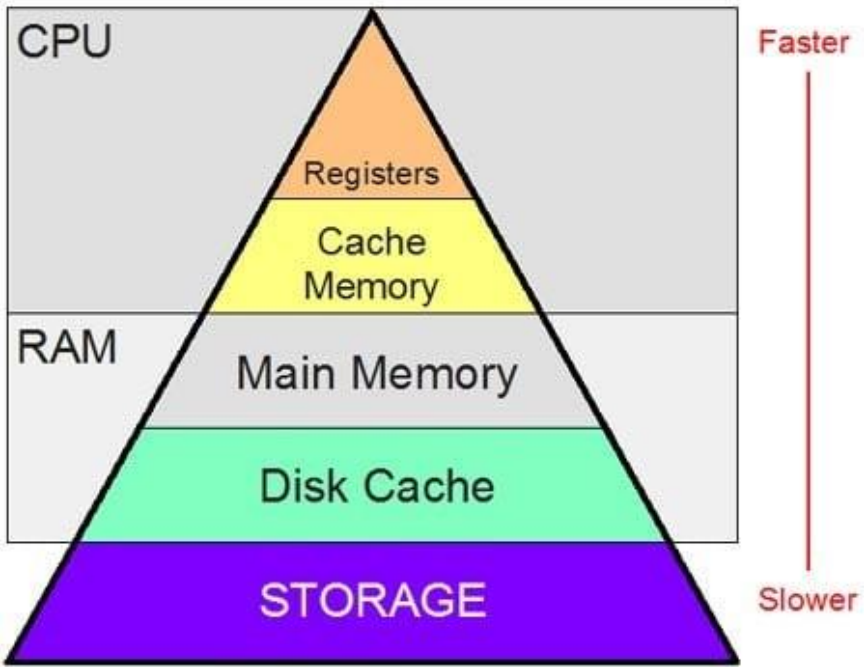

Рисунок 2.1 – Процес збереження даних у пристроях комп’ютеру

Для ефективного керування великими обсягами даних комп’ютери використовують файлові системи.

Файлова система - це ієрархічна структура, яка організовує дані у файли та папки. Файли - це окремі одиниці даних, які містять певну інформацію, наприклад документ, зображення чи відео. Папки - це контейнери, які містять кілька файлів і можуть бути вкладені в інші папки для створення ієрархічної організації. Файлові системи відіграють вирішальну роль в управлінні тим, як дані зберігаються, організовуються та витягуються на пристроях зберігання. Різні операційні системи використовують різні файлові системи, кожна з яких має свої особливості. Файлові системи поділяються на:

FAT16, FAT32, exFAT - використовуються переважно в середовищах Windows. FAT використовує таблицю розміщення файлів, щоб відстежувати, де на диску зберігається кожен фрагмент файлу. Проте, FAT має обмеження щодо розміру файлу та розміру розділу, що робить його менш придатним для великих файлів або сучасних пристроїв зберігання даних.

NTFS (файлова система нової технології) - є файловою системою за замовчуванням для операційних систем Windows. Підтримує списки контролю доступу (ACL), шифрування та стиснення. Він забезпечує кращу безпеку та надійність порівняно з файловими системами FAT. NTFS підтримує більші розміри файлів і розділів порівняно з FAT.

ExFAT (розширена таблиця розміщення файлів) - розроблено як файлову систему, яка добре працює в різних операційних системах. Усуває деякі обмеження файлових систем FAT, допускаючи більші розміри файлів і розділи. Часто використовується на флеш-накопичувачах і зовнішніх пристроях зберігання даних.

HFS+ (ієрархічна файлова система плюс) і APFS (файлова система Apple), яка використовується в macOS. HFS+ була традиційною файловою системою, тоді як APFS є новою, вдосконаленою файловою системою. APFS розроблено для сучасних технологій зберігання, таких як SSD. Він підтримує такі функції, як знімки, клонування та спільне використання простору.

2.2 Аналіз захищених методів збереження даних

Безпека зберігання даних передбачає захист ресурсів зберігання та даних, що зберігаються – як локальних, так і зовнішніх центрах обробки даних і хмари – від випадкового чи навмисного пошкодження чи знищення, а також від неавторизованих користувачів і використання[18]. Це сфера, яка має вирішальне значення для підприємств, оскільки більшість порушень даних зрештою спричинені збоєм безпеки зберігання даних. Захищене зберігання даних у сукупності відноситься до ручних і автоматизованих обчислювальних процесів і технологій, які використовуються для забезпечення безпеки та цілісності збережених даних. Це може включати фізичний захист апаратного забезпечення, на якому зберігаються дані, а також програмне забезпечення для забезпечення безпеки. Безпечне зберігання даних стосується тих даних, які зберігаються на жорстких дисках комп'ютера/сервера, портативних пристроях, таких як зовнішні жорсткі диски або USB-накопичувачі, а також онлайн/хмара, мережеве сховище (SAN - Storage Area Network) або мережеве сховище (NAS - Network Attached Storage) системи.

Безпечне зберігання даних досягається за рахунок:

- Шифрування даних;
- Механізм контролю доступу до кожного пристрою зберігання даних/програмного забезпечення;
- Захист від вірусів, хробаків та інших загроз пошкодження даних;
- Безпека фізичного/пілотованого пристрою зберігання та інфраструктури;
- Забезпечення та реалізація багаторівневої архітектури безпеки зберігання.

Безпечне зберігання даних має важливе значення для організацій, які мають справу з конфіденційними даними, як для того, щоб уникнути крадіжки даних, так і для забезпечення безперебійної роботи.

Безпека зберігання та безпека даних тісно пов'язані із захистом даних. Безпека даних передусім передбачає збереження приватної інформації від рук тих, хто не має права переглядати її. Він також включає захист даних від інших типів атак, наприклад

програм-вимагачів, які перешкоджають доступу до інформації, або атак, які змінюють дані, роблячи їх ненадійними.

Захист даних – це більше гарантія того, що дані залишаються доступними після менш неприємних інцидентів, таких як збої системи чи компонентів або навіть стихійні лиха. Але вони перетинаються через спільну потребу забезпечити надійність і доступність інформації, а також у необхідності відновлення після будь-яких інцидентів, які можуть загрожувати даним.

Загрози можна розділити на дві категорії: зовнішні та внутрішні. До зовнішніх загроз належать хакери, кіберзлочинці, організовані злочинні групи та конкуренти, які здійснюють “промислове шпигунство”. Внутрішні загрози включають погано навчений або недбалий персонал, незадоволених працівників. Серед інших загроз пожежі, повені та інші стихійні лиха, відключення електроенергії.

Ще одним величезним фактором у безпеці зберігання даних є вразливі місця, притаманні системам зберігання. Вони включають:

- Відсутність шифрування - хоча деякі високоякісні пристрої NAS і SAN включають автоматичне шифрування, багато продуктів на ринку не включають ці можливості. Це означає, що користувачам потрібно встановити окреме програмне забезпечення або пристрій для шифрування, щоб переконатися, що їхні дані зашифровані;
- Використання хмарних сховищ - все більше підприємств вирішують зберігати частину або всі свої дані в хмарі. Хоча існує думка, що хмарне сховище безпечніше, ніж локальне сховище, хмара ускладнює середовище зберігання та часто вимагає від персоналу сховищ вивчати нові інструменти та володіти глибокими знаннями конкретного хмарного сховища, щоб гарантувати належний захист даних;
- Неповне знищення даних - коли дані видаляються з жорсткого диска чи іншого носія, вони можуть залишити сліди, які можуть дозволити неавторизованим особам відновити цю інформацію. Адміністратори та менеджери сховища повинні переконатися, що будь-які дані, стерті зі сховища, перезаписуються, щоб їх неможливо було відновити;

- Відсутність фізичної безпеки - деякі організації не приділяють належної уваги фізичній безпеці своїх пристроїв зберігання. У деяких випадках вони не враховують, що внутрішня особа, як-от співробітник або член бригади з прибирання, може отримати доступ до фізичних пристроїв зберігання даних і отримати дані, минаючи всі ретельно сплановані заходи безпеки на основі мережі.

Безпека зберігання даних прагне забезпечити конфіденційність, цілісність і доступність.

Збереження конфіденційності даних досягається шляхом забезпечення того, що до них не можуть отримати доступ неавторизовані особи через мережу чи локально, є ключовим принципом безпеки зберігання для запобігання витоку даних.

Цілісність даних у контексті безпеки зберігання даних означає гарантію того, що дані не можуть бути підроблені або змінені.

У контексті безпеки зберігання даних доступність означає мінімізацію ризику того, що ресурси зберігання будуть знищені або недоступні навмисно – скажімо, під час DDoS-атаки – або випадково через стихійне лихо, збій живлення чи механічну поломку.

Для безпечного зберігання даних рекомендовано застосувати наведені далі методи безпеки даних.

Політики безпеки зберігання даних, що визначають відповідні рівні безпеки для різних типів даних. Очевидно, що загальнодоступні дані потребують значно меншої безпеки, ніж дані з обмеженим доступом або конфіденційні дані. Організації повинні мати моделі безпеки, процедури та інструменти для застосування належного захисту. Політики безпеки також повинні включати детальну інформацію про заходи безпеки, які слід застосовувати на пристроях зберігання даних, які використовуються організацією.

Контроль доступу на основі ролей є обов’язковим для захищеної системи зберігання даних, і в деяких випадках може бути доцільною багатофакторна автентифікація. Адміністратори також мають переконатися, що змінили будь-які

паролі за замовчуванням на своїх пристроях зберігання даних і змусили користувачів використовувати надійні паролі.

Шифрування - дані повинні бути зашифровані як під час передачі, так і під час спокою в системах зберігання. Адміністратори сховищ також повинні мати безпечні системи керування ключами для відстеження своїх ключів шифрування.

Запобігання втраті даних - багато експертів кажуть, що лише шифрування недостатньо для повної безпеки даних. Вони рекомендують організаціям також розгортати рішення для запобігання втраті даних (DLP), які можуть допомогти знайти та зупинити будь-які поточні атаки.

Сильна мережева безпека - системи зберігання не існують у вакуумі, вони повинні бути оточені потужними системами мережевої безпеки, такими як брандмауери, захист від зловмисного програмного забезпечення, шлюзи безпеки, системи виявлення вторгнень і, можливо, передові рішення безпеки на основі аналітики та машинного навчання. Ці заходи мають запобігти більшості кібератак від отримання доступу до пристроїв зберігання даних.

Надійна безпека на пристроях кінцевих користувачів. Крім того, організації також повинні переконатися, що вони мають відповідні заходи безпеки на ПК, смартфонах та інших пристроях, які матимуть доступ до збережених даних. Ці пристрої, можуть бути слабким місцем у системі кіберзахисту організації.

Надлишковість - надлишкове сховище, включаючи технологію RAID, не тільки допомагає підвищити доступність і продуктивність, а в деяких випадках також може допомогти організаціям пом’якшити інциденти безпеки.

Резервне копіювання та відновлення. Деякі успішні атаки зловмисного програмного забезпечення або програм-вимагачів настільки повно компрометують корпоративні мережі, що єдиним способом відновлення є відновлення з резервних копій. Менеджери сховищ мають переконатися, що їхні системи та процеси резервного копіювання відповідають таким подіям, а також цілям аварійного відновлення. Крім того, вони повинні переконатися, що системи резервного копіювання мають такий же рівень безпеки даних, як і основні системи.

Існують протоколи, які забезпечують безпеку зберігання та передавання даних. Серед них IPFS (InterPlanetary File System): IPFS - це протокол та мережа для децентралізованого зберігання та обміну гіпертекстового змісту. Файли в IPFS ідентифікуються за допомогою унікальних хеш-сум, а не URL. Це дозволяє зберігати файли децентралізовано та ефективно розподіляти їх за допомогою Peer-to-Peer мережі.

Ethereum блокчейн: Ethereum - це платформа для створення розумних контрактів і децентралізованих додатків на блокчейні. Ethereum має свої власні токени (ETH), які можуть використовуватися для розрахунків у розумних контрактах.

Модель для зберігання файлів на базі блокчейн може виглядати так:

IPFS для зберігання файлів - файли завантажуються на IPFS, і кожен файл отримує свою унікальну хеш-суму, яка служить як посилання на цей файл.

Розумні контракти Ethereum створюються розумні контракти на платформі Ethereum, які дозволяють управляти доступом до файлів та відслідковувати належність файлів.

Збереження хеш-суми у блокчейні розумний контракт Ethereum може містити вказівник на хеш-суму файлу на IPFS та іншу інформацію, таку як права доступу та власник файлу.

Управління доступом - за допомогою розумних контрактів Ethereum, можна встановлювати права доступу до файлів для різних користувачів. Тільки користувачі з відповідними дозволами зможуть отримати доступ до файлів.

Журналізація історії змін - блокчейн Ethereum зберігає записи про всі зміни у розумних контрактах, що дозволяє вам відстежувати історію змін доступу до файлів.

Така модель забезпечує децентралізоване зберігання файлів, високий рівень безпеки та можливість відстеження доступу до файлів. Однак слід враховувати, що використання Ethereum може вимагати плати за транзакції, що може збільшувати витрати на зберігання файлів.

Для захисту даних можна застосувати маскування даних[19]. Метод маскування даних передбачає модифікацію вихідних даних у спосіб, що робить їх невпізнанними. У результаті можна створити неавтентичну версію набору конфіденційних даних, яка

структурно схожа на справжні дані. Не маючи доступу до вихідного набору даних, неможливо виконати зворотне проєктування або відкат до вихідних значень даних. Застосовуючи маскування даних, можна зберігати конфіденційну корпоративну інформацію та конфіденційні записи клієнтів для різних цілей, не підвищуючи ризики для безпеки чи відповідності.

Загалом маскування даних допомагає створювати чіткі політики захисту даних і гарантувати, що найважливіші записи добре захищені.

Цей метод захисту даних також можна легко застосувати до всіх типів активів даних. Наприклад, можна виконати маскування даних у локальній базі даних, а потім перенести замасковану копію даних у стороннє середовище. Такий підхід називається статичним маскуванням даних.

Крім того, можна використовувати динамічне маскування даних (DDM), щоб уникнути створення зайвих копій даних. У цьому випадку дані залишаються розкритими в локальній базі даних, але анонімізуються на вимогу кожного разу, коли надходить запит на спільний доступ (наприклад, запит на завантаження в певне сховище даних).

Останній підхід полягає в маскуванні даних на льоту, тобто в режимі реального часу, коли дані переміщуються між двома середовищами. Маскування даних на льоту є найдорожчим у підтримці, але це важлива інвестиція, якщо потрібна постійна інтеграція або синхронізація різнорідних наборів даних.

Впровадження шифрування даних передбачає кодування інформації таким чином, що її може розшифрувати лише сторона, яка має доступ до ключа шифрування. Шифрування є поширеним методом захисту.

Одним із найвідоміших методів шифрування є протокол Secure Sockets Layer (SSL), який створює безпечне з’єднання між веб-сервером і веб-браузером для забезпечення безпечного обміну наданими деталями (наприклад, інформацією про вхід/пароль). Більшість із нас знають SSL як з’єднання https://.

Два найпоширеніші типи шифрування даних - це симетричне та асиметричне шифрування.

Симетричне шифрування[20] використовує той самий ключ для шифрування та дешифрування даних. Такі спільні ключі вимагають менше обчислювальної потужності для роботи та є більш економічно ефективними в обслуговуванні. Декодування даних також відбувається з більшою швидкістю. Недоліком є те, що якщо спільний ключ потрапить у чужі руки, безпека системи буде порушена. Принципи симетричного шифрування відображає рисунок 2.2.

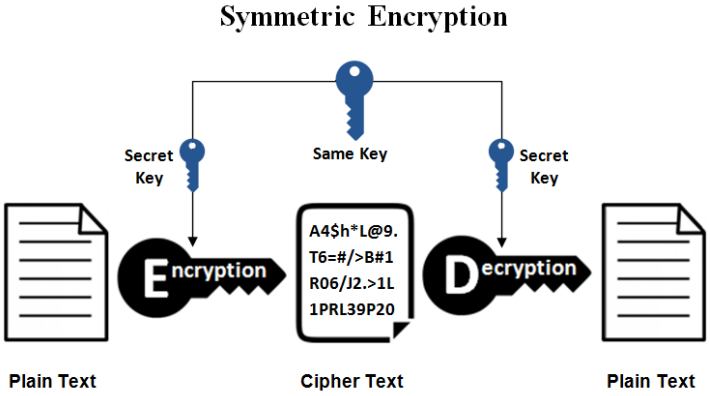

Рисунок 2.2 – Симетричне шифрування

Асиметричне шифрування[20] використовує два окремих ключі для шифрування даних. Відкритий ключ використовується всіма для надсилання зашифрованих даних, а закритий ключ одна сторона може використовувати для розшифровки повідомлення. Це вимогливіший метод шифрування, який вимагає більшої обчислювальної потужності та може бути не найкращим вибором для надсилання великих пакетів даних. Принципи асиметричного шифрування продемонстровано на рисунку 2.3.

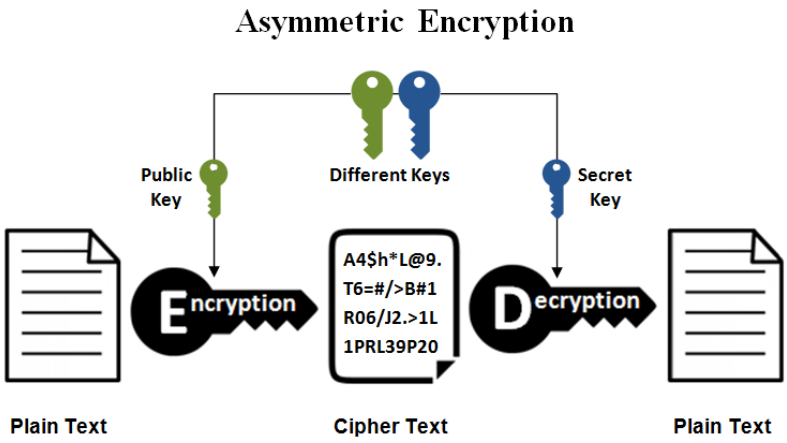

Рисунок 2.3 – Асиметричне шифрування

Для забезпечення безпеки даних було обрано популярні алгоритми шифрування AES[11] та ChaCha20[10].

Шифрування AES використовує симетричний блоковий шифр – це алгоритм шифрування, розроблений Національним інститутом стандартів і технологій (NIST) у 1997 році, щоб зробити урядові дані менш вразливими до атак.

Алгоритм AES[11] розбиває повідомлення на менші блоки. Крім того, замість одного раунду шифрування існує кілька, включаючи заміну, транспозицію та змішування. 256-бітний ключ використовує 14 раундів шифрування.

AES – це симетричний блоковий шифр, він використовує той самий ключ (або шифр) для шифрування та дешифрування даних. Однак алгоритм шифрування AES значно відрізняється від стандартного симетричного шифрування. Крім початкового шифрування повідомлення меншими блоками, а не всіма відразу, подальші раунди шифрування роблять повідомлення ще важчим для декодування.

Кожен раунд алгоритму AES[21] складається з чотирьох кроків:

Підстановка: алгоритм замінює звичайний текст на зашифрований текст на основі попередньо визначеного шифру.

Зміщення: всі рядки зсуваються на одиницю, крім першого.

Змішування: інший шифр, званий шифром Хілла, використовується для змішування стовпців, щоб хтось не міг просто зрушити рядки назад, щоб почати розшифровку даних.

Подальше шифрування: для шифрування цього блоку даних використовується невелика частина ключа шифрування.

Зі збільшенням розміру ключа зростає складність злому шифрування. Незалежно від розміру ключа, експерти вважають AES безпечним проти атак грубою силою. Через кілька рівнів шифрування сучасні комп’ютери не можуть зламати AES за розумний проміжок часу.

Розшифровка AES проста і по суті така ж, як шифрування AES, але навпаки. Одержувач зашифрованих даних також має копію використаного шифру, тому вони виконують кожну функцію у зворотному порядку, щоб видалити шари шифрування.

Інформація надсилається в зашифрованому вигляді через Інтернет, де ключ AES використовується для розшифровки даних у звичайний текст, який читає одержувач.

Алгоритм AES спочатку був розроблений NIST для державних потреб, але зараз використовується як у державних, так і в приватних програмах. VPN, менеджери паролів, мобільні програми, бездротові мережі, шифрування файлів і навіть відеоігри використовують шифрування AES,

ChaCha20[10] – це симетричний алгоритм шифрування, який використовує 256-бітний ключ як для шифрування, так і для дешифрування.

Він був розроблений Деніелом Дж. Бернштейном, відомим криптографом, у 2008 році як потоковий шифр. Алгоритм шифрування ChaCha20 розроблений для забезпечення поєднання швидкості та безпеки.

Даний алгоритм створений для захисту від відомих атак, зокрема диференційного та лінійного крипто аналізу. Крім того, він добре розпаралелений, що робить його легко адаптованим до багатоядерних процесорів та інших високопродуктивних обчислювальних систем.

ChaCha20 — це потоковий шифр, тобто він шифрує дані безперервним потоком, а не блоками фіксованого розміру.

Він генерує безперервний потік ключів із псевдовипадкових бітів, які потім об’єднуються XOR з даними відкритого тексту для створення зашифрованого тексту.

Основні етапи процесу шифрування ChaCha20:

Генерація ключа: Алгоритм ChaCha20 генерує 256-бітний ключ із ключа, наданого користувачем, і випадково згенерованого 96-бітного одноразового коду.

Ініціалізація: алгоритм використовує ключ і nonce для ініціалізації стану шифру.

Шифрування даних: ChaCha20 шифрує кожен блок даних, використовуючи стан шифру, який оновлюється після обробки кожного блоку.

Вихід: результуючий зашифрований текст створюється шляхом операції XOR відкритого тексту з результатом кроку шифрування даних.

Шифрування ChaCha20 пропонує кілька переваг над іншими алгоритмами шифрування, зокрема:

1. Швидкість: ChaCha20 є одним із найшвидших доступних алгоритмів шифрування, придатних для ефективного застосування на різних пристроях, включаючи мобільні пристрої та пристрої IoT з низьким енергоспоживанням.

2. Безпека: ChaCha20 розроблений для високої безпеки, стійкий до відомих атак. Він заснований на принципах, подібних до алгоритму шифрування Salsa20, ретельно протестований і вважається дуже безпечним.

3. Розпаралелення: ChaCha20 має високу можливість розпаралелювання, що робить його судовим вибором для багатоядерних процесорів і високопродуктивних обчислювальних систем.

4. Проста реалізація. Порівняно з іншими алгоритмами шифрування, ChaCha20 відносно легко реалізувати, що робить його привабливим вибором для розробників.

Шифрування ChaCha20 застосовується в різноманітних програмах, зокрема:

1. Безпечний зв’язок: ChaCha20 захищає зв’язок між сторонами в безпечних програмах обміну повідомленнями або VPN.

2. Шифрування файлів: шифрує файли, що зберігаються на пристрої або передаються через мережу.

3. Безпека IoT: ChaCha20 захищає пристрої IoT, які часто мають обмежені обчислювальні ресурси та потребують легких алгоритмів шифрування.

4. Веб-безпека: його можна використовувати для захисту веб-трафіку, наприклад з’єднань HTTPS.

Шифрування ChaCha20 — це швидкий і надійний алгоритм, який пропонує численні переваги для різних програм. Його простота, швидкість і безпека роблять його чудовим вибором для розробників, яким потрібно захистити конфіденційні дані. Щоб максимально підвищити безпеку, дуже важливо дотримуватися надійних методів керування ключами та найкращих методів безпечного шифрування.

2.3 Вибір технологій та середовища розробки

Наразі існує багато фреймворків, бібліотек, технологій, мов програмування та допоміжних засобів, у тому числі із використанням штучного інтелекту, які можуть використовуватися при розробці програмного продукту. Буває, що з'являються нові технології, а старі перестають підтримуватися розробниками та спільнотою, у зв'язку із цим варто звертати увагу на терміни підтримки технології компаніями. Зважаючи на вищеперераховані фактори, при виборі технологій для розробки сховища даних із використанням технологій блокчейн, було вирішено звертати увагу на такі ознаки:

- Технологія, яка доступна для безкоштовного використання;
- Відкритий вихідний код (open source);
- Наявність великого ком'юніті, яке розробляє додатки із використанням технології(дозволить легко знаходити вирішення типових проблем);
- Підтримка обраних технологій блокчейн, або наявність відповідних бібліотек;
- Можливість створення додатків серйозного рівня;
- Кросплатформність.

Данні критерії задовольняє платформа dotnet, із використанням мови програмування C# для написання коду.

.NET[6] –популярна технологія, яка активно розвивається за підтримки користувачів та компанії Microsoft. Вона дозволяє створювати як як прості додатки для власного використання так і серйозні комерційні. Окрім цього, доступно безліч бібліотек, які дозволяють вирішити базові задачі.

Платформа dotnet має відкритий вихідний код та підтримується компанією Microsoft. Вона бере на себе всю роботу із виділення та звільненням пам'яті, існує продвинутий збирач сміття, який дозволяє не думати про контроль пам'яті, а зосередитися на написанні коду, допускаючи менше помилок.

Для написання коду було обрано мову програмування C#[6], яка має лаконічний, сі-подібний синтаксис, регулярно оновлюється та є строго типізованою, об'єктно орієнтовною. У якості середовища розробки було обрано Visual studio 2022

community, яка є безкоштовною, але включає увесь необхідний для розробки та відладки функціонал серед якого:

- Можливість створювати шаблони для типових проектів;
- Наявність зручного меню відлагодження програми;
- Контроль використання пам’яті та процесора;
- Інтегрована система контролю версій.

Для створення фронтенд частини доцільно обрати Angular, який у парі з typescript, який є строго типізованою покращеною версією стандартного JavaScript, дозволяє пришвидшити процес розробки шляхом винесення коду у сервіси та модулі. А використанням типів дозволяє відловити певні помилки на етапі компіляції та зменшує ймовірність непередбачуваної поведінки, на відміну від чистого JavaScript. Angular використовує:

- HTML - для створення основної структури веб-сторінок. HTML визначає елементи, такі як заголовки, абзаци, таблиці, які формують основний контент сторінки;
- CSS - для стилізації веб-сторінок. CSS дозволяє змінювати вигляд та відображення елементів HTML, таких як кольори, розміри та розташування, для створення привабливого та естетичного дизайну;
- Typescript - для додавання інтерактивності до веб-сторінок. JavaScript дозволяє вам створювати динамічні ефекти, обробляти події користувача та взаємодіяти з сервером без перезавантаження сторінки.

Для обробки даних було вирішено використовувати базу даних. За методом зберігання бази даних можна умовно розділити на дві категорії - SQL та NoSQL. SQL має чітко задану структуру таблиць, а NoSQL бази зберігають дані без чіткої структури та без зв'язків між ними. Це означає, що ми можемо зберігати різні дані, такі як JSON-об'єкти, числа, аудіо файли та інші, без обмежень.

SQL[5] бази даних, завдяки строго визначеним моделям та зв'язкам, забезпечують цілісність даних та спрощують логіку роботи з даними при наявності чітко визначених моделей предметної області. З урахуванням важливого фактору - цілодобової доступності та стабільності бази даних - багато провайдерів послуг

переходять на хмарні сервіси. Зокрема, SQL server, який доступний для безкоштовного використання і надає надійні та швидкі методи роботи із даними будь яких об’ємів. Із часом можна перейти від безкоштовної версії до платної та отримати доступ до продвинутих функцій.

Для виконання операцій над даними із програмного продукту, було вирішено обрати технологію Entity Framework Core[12]. Ця технологія дозволяє абстрагуватися від постачальників баз даних та винести логіку роботи із даними в об’єктно орієнтований стиль, із використанням класів. Завдяки цій технології можна будувати бізнес-логіку, незалежно від конкретної бази даних. Крім того, Entity Framework Core дозволяє абстрагуватися від використання конкретних SQL запитів і працювати з даними за допомогою інструментів середовища .Net, зокрема, мови програмування C#.

2.4 Висновки

Інформація – цінний ресурс, який по фрагментах збирається у файли, які можуть групуватися у папки та зберігатися у файлових системах, які у свою чергу розгорнуті на пристроях збереження даних. У процесі захищеного зберігання даних важливо знайти компроміс між безпекою та швидкодією, навіть, якщо існує алгоритм, який неможливо зламати сьогодні – часи можуть змінитися, і завтра він стане ненадійним. Цю проблему може вирішити використання каскадного шифрування. Безпека зберігання даних є одним із основних елементів високої зрілості кібербезпеки. Контроль доступу, регулярне резервне копіювання та застосування методів захисту даних, таких як маскування та шифрування даних, є важливими для запобігання втраті даних, корпоративним зломам та іншим кіберінцидентам. Впровадження даних засобів вимагає використання певних технологій, оптимальним вибором є строго типізована, об’єктно орієнтована мова програмування із відкритим вихідним кодом та великим ком’юніті – C# у поєднанні із платформою dotnet для розробки web API, та Angular для розробки веб клієнту.

3. АНАЛІЗ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ЗАХИЩЕНОГО СХОВИЩА ДАНИХ

3.1 Опис алгоритму захищеного збереження даних з використанням технологій блокчейн

Для забезпечення безпеки конфіденційної інформації, було вирішено використати комбінацію каскадного шифрування – для захисту даних від крадіжки або неправомірного доступу та технології блокчейн для захисту від підміни даних у файлі.

Каскадне шифрування відноситься до методу шифрування, в якому дані проходять через послідовність двох або більше алгоритмів шифрування. Кожен етап використовує свій власний ключ і власний алгоритм шифрування. Такий підхід може підвищити безпеку шифрування, оскільки для того, щоб отримати доступ до файлу необхідно обійти не один, а декілька послідовних алгоритмів.

Принцип каскадного шифрування може бути ілюстрований наступним чином:

Етап 1: шифрування Алгоритмом 1 з Ключем 1. Перші дані шифруються за допомогою першого алгоритму з використанням ключа 1.

Етап 2: шифрування Алгоритмом 2 з Ключем 2. Зашифровані дані з етапу 1 подаються на вхід другого алгоритму, який використовує другий ключ для шифрування і так далі.

Процес може продовжуватися каскадно, використовуючи більше етапів шифрування.

Підвищення безпеки виникає тому, що навіть якщо один із ключів або алгоритмів буде зламано, проникнення в систему не надасть зловмисникам можливості легко розшифрувати дані. Кожен етап у каскаді має власні ключі і параметри, що робить ситуацію складнішою для атакуючих. Однак, важливо враховувати, що додавання багатьох етапів каскадного шифрування може призвести до збільшення обчислювальних витрат і затримки при обробці даних.

Для забезпечення захисту від підміни даних у файлі було використано наступний алгоритм.

Збереження хешу[22] файлу. Спочатку користувач завантажує файл на систему. Після завантаження система автоматично обчислює хеш-суму цього файлу за допомогою криптографічних хеш-функцій, таких як SHA-256. Отриманий хеш-код потім записується в блокчейн. Блокчейн використовується для забезпечення недоступності зміни хешу, що дозволяє користувачу перевірити цілісність файлу у будь-який момент. Блоки з хешами об'єднуються у ланцюг, забезпечуючи безпеку та надійність збереження інформації. Блок схему алгоритму шифрування файлів із використанням технології блокчейн та каскадного шифрування зображено у додатку Б на рисунку Б.1.

Завдяки цьому двоетапному алгоритму захищеного збереження даних, користувачі можуть бути впевнені в безпеці та конфіденційності своєї інформації. Кожна взаємодія із файлами буде зафіксована в блокчейні, а шифрування гарантує, що навіть при потенційному порушенні безпеки, важлива інформація залишається недоступною для зловмисників.

3.2 Проєктування архітектури додатку

Архітектура додатку описує шаблони та методи, які використовуються для проєктування та створення програми. Архітектура це найкращі практики, яких слід дотримуватися під час створення програми, щоб отримати добре структуровану програму. Продумана архітектура дозволяє полегшити процес виправлення помилок та процес подальшого розвитку продукту.

Як частина архітектури програми, існують як зовнішні, так і внутрішні служби. Розробка клієнтської частини пов’язана з користувальницьким досвідом роботи програми, а бекенд розробка зосереджена на наданні доступу до даних, бізнес логіки та інших існуючих систем, які забезпечують роботу програми.

Історично склалося так, що програми писалися як єдиний код, де всі компоненти спільно використовують однакові ресурси та простір пам’яті. Цей стиль архітектури називають монолітним. Сучасні архітектури додатків частіше слабко пов’язані, використовуючи мікросервіси та прикладний програмний інтерфейс (API).

Існує багато різних типів архітектур додатків, але найпоширенішими є:

- моноліт[23]
- n-рівнева архітектура
- мікросервіси[23]
- архітектура, керована подіями
- архітектура, орієнтована на послуги.

Для порівняння було обрано дві найпопулярніші архітектури: монолітну та мікросервісну.

Монолітна архітектура[24] - це традиційна модель програмного забезпечення, яка побудована як уніфікована одиниця, автономна та незалежна від інших програм. Слово «моноліт» часто пов’язують із чимось великим і льодовиковим, що не так далеко від істини монолітної архітектури для розробки програмного забезпечення. Монолітна архітектура - це окрема велика обчислювальна мережа з єдиною кодовою базою, яка об’єднує всі бізнес-завдання. Щоб внести зміни в програму такого типу, потрібно оновити весь проект, отримавши доступ до бази коду та створивши і розгорнувши оновлену версію додатку. Це робить оновлення обмежувальними та забирає багато часу. Моноліти можуть бути зручними на ранніх стадіях життя проекту для полегшення керування кодом, витрат на підтримання і розгортання.

Приклад монолітної архітектури зображено на рисунку 3.1

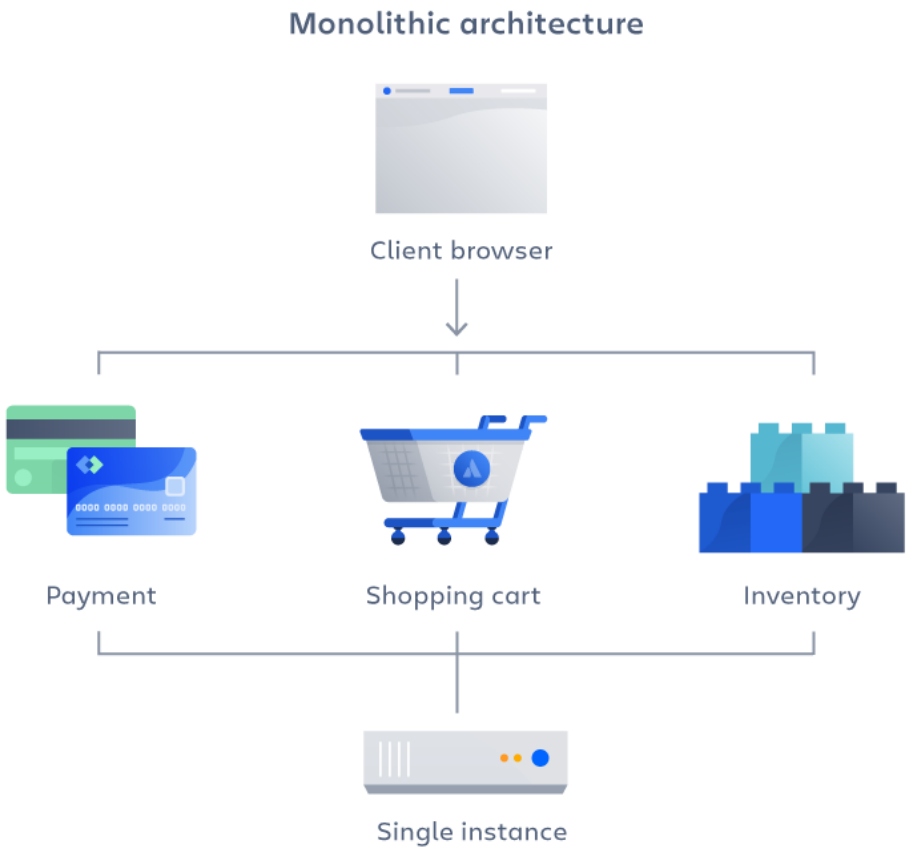

Рисунок 3.1 – Приклад монолітної архітектури додатку

До переваг монолітної архітектури можна віднести:

- Просте розгортання – один виконуваний файл або каталог спрощує розгортання.
- Розробка – коли програму створено в одному репозиторії, її легше розробляти.
- Продуктивність – у централізованій базі коду та сховищах, один API часто може виконувати ту саму функцію, яку виконують численні API з мікросервісами.
- Спрощене тестування. Оскільки монолітна програма є єдиним централізованим блоком, наскрізне тестування можна виконати швидше, ніж із розподіленою програмою.

- Легке налагодження. Завдяки тому, що весь код розміщено в одному місці, легше виконати запит і знайти проблему.

Монолітні програми можуть бути досить ефективними, поки вони не стануть занадто великими і масштабування не стане проблемою. Внесення невеликих змін до однієї функції вимагатиме компіляції та тестування всієї платформи, що суперечить гнучкому підходу, якому надають перевагу розробники.

До недоліків моноліту можна віднести:

- Повільніша швидкість розробки – велика, монолітна архітектура робить розробку складнішою та повільнішою;
- Масштабованість – немає можливості масштабувати окремі компоненти.
- Надійність - якщо в будь-якому модулі є помилка, це може вплинути на доступність усієї програми;
- Перешкода для впровадження технології. Будь-які зміни у структурі чи мові впливають на всю програму, роблячи зміни часто дорогими та трудомісткими;
- Відсутність гнучкості – моноліт обмежений технологіями, які вже використовуються в моноліті;
- Розгортання – невелика зміна монолітної програми вимагає повторного розгортання всього моноліту.

Мікросервісна архітектура, також відома як мікросервіси - це архітектурний метод, який базується на ряді незалежно розгорнутих сервісів. Ці сервіси мають власну бізнес-логіку та базу даних. Оновлення, тестування, розгортання та масштабування відбуваються в межах кожної служби. Мікросервіси відокремлюють основні бізнес-специфічні завдання на окремі незалежні кодові бази. Мікросервіси не зменшують складність, але вони роблять будь-яку складність видимою та більш керованою, розділяючи завдання на менші процеси, які функціонують незалежно один від одного. Запровадження мікросервісів тісно пов’язані з DevOps, це є основою для безперервної доставки оновлень, які дозволяють командам швидко адаптуватися до вимог користувачів.

Приклад мікросервісної архітектури зображено на рисунку 3.2.

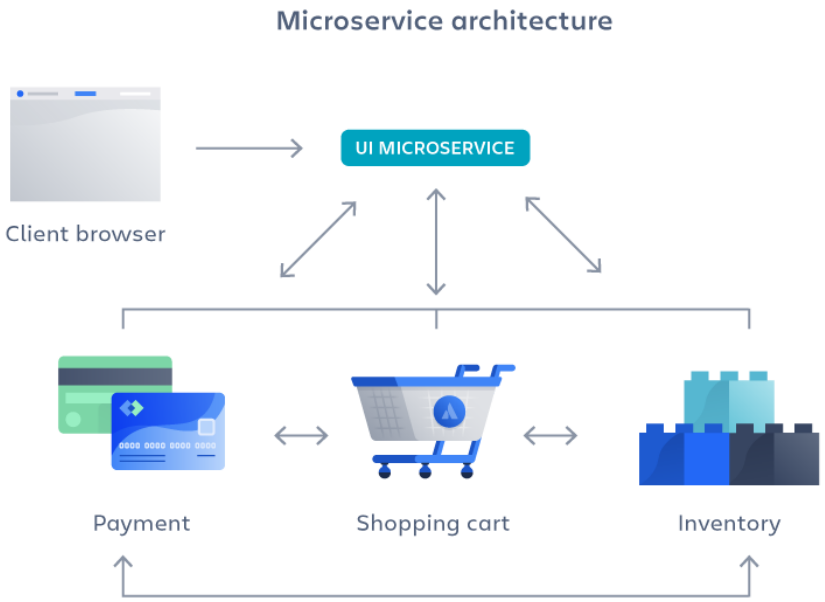

Рисунок 3.2 – Приклад мікросервісної архітектури

Мікросервіси аж ніяк не є ідеальними, але вони вирішують низку проблем для програмного забезпечення та продуктів, що активно розвиваються. Оскільки архітектура мікросервісів складається з сутностей, які працюють незалежно, кожну сутність можна розробляти, оновлювати, розгортати та масштабувати, не впливаючи на інші служби. Це дозволяє частіше випускати оновлення та розгортати їх на сервері не впливаючи на вже існуючі версії ресурсів. Оновлення програмного забезпечення можна виконувати частіше, з покращеною надійністю, часом безвідмовної роботи та продуктивністю.

Переваги мікросервісів такі:

- Гнучкість – сприяє гнучким способам роботи з невеликими командами, які часто випускають оновлення;
- Гнучке масштабування – якщо мікросервіс досягає високого навантаження, нові екземпляри цього сервісу можна швидко розгортати в супровідному кластері, щоб зменшити навантаження;
- Безперервне розгортання – дозволяє отримати часті та швидші цикли випусків;

- Зручність обслуговування та тестування – команди можуть експериментувати з новими функціями та повертатися, якщо щось не працює. Це полегшує оновлення коду та пришвидшує час виходу на ринок нових функцій. Крім того, легко виявляти та виправляти несправності та помилки в окремих службах;
- Незалежне розгортання – оскільки мікросервіси є окремими одиницями, вони дозволяють швидко та легко і незалежно розгортати окремі функції;
- Технологічна гнучкість – архітектури мікросервісів дають командам свободу вибору інструментів, які вони бажають;
- Висока надійність – можна розгортати зміни для конкретної служби без загрози виходу з ладу всієї програми;

До недоліків мікросервісів можна віднести:

- Ускладнення розробки – мікросервіси додають більше складності порівняно з монолітною архітектурою, оскільки є більше сервісів у багатьох місцях, створених декількома командами. Якщо розробка не управляється належним чином, це призводить до уповільнення швидкості розробки та низької продуктивності;
- Експоненціальні витрати на інфраструктуру. Кожна нова мікрослужба може мати власну вартість набір тестів, посібники з розгортання, інфраструктуру хостингу, інструменти моніторингу тощо;
- Додаткові організаційні витрати – командам потрібно додати інший рівень спілкування та співпраці, щоб координувати оновлення та інтерфейси;
- Проблеми з налаштуванням. Кожен мікросервіс має власний набір журналів, що ускладнює налаштування. Крім того, один бізнес-процес може працювати на кількох машинах, що ще більше ускладнює його налаштуванням;
- Відсутність стандартизації – без спільної платформи можливе використанням різних мов, технологій та платформ, а також стандартів до написання коду;
- Відсутність чіткої приналежності – із запровадженням більшої кількості послуг зростає й кількість команд, які керують цими послугами. З часом стає важко знати, якими послугами може скористатися команда та до кого звернутися за підтримкою.

В результаті для розробки системи було вирішено обрати монолітну архітектуру, проте із використанням грамотного планування та розбиття файлів по папках. Використання найкращих підходів і створення окремих моделей для бізнес логіки, презентації та роботи із даними, в майбутньому це дозволить із легкістю перейти на мікросервісну архітектуру, за потреби. Також цей підхід надасть необхідну легкість та швидкість розробки на початкових етапах(переваги монолітної архітектури), а в майбутньому за потреби масштабування чи делегування розробки іншим командам чи необхідності використання нових технологій використати переваги мікросервісів.

3.3 Розробка UML діаграм роботи системи

UML[26], скорочення від Unified Modeling Language, використовується для візуалізації програмного моделювання за допомогою набору діаграм. Це не тільки допомагає оцінити всю концепцію проекту, але й гарантує, що всі в команді матимуть розуміння щодо структури та поведінки проекту. Діаграма UML також є одним із популярних методів моделювання бізнес-процесів. Це об’єктно-орієнтована схема представлення, яка описує зв’язок між акторами та системами. Існують різні типи UML, такі як діаграми варіантів використання, діаграми послідовності, діаграми зв’язку тощо.

Види UML діаграм зображено на рисунку 3.3.

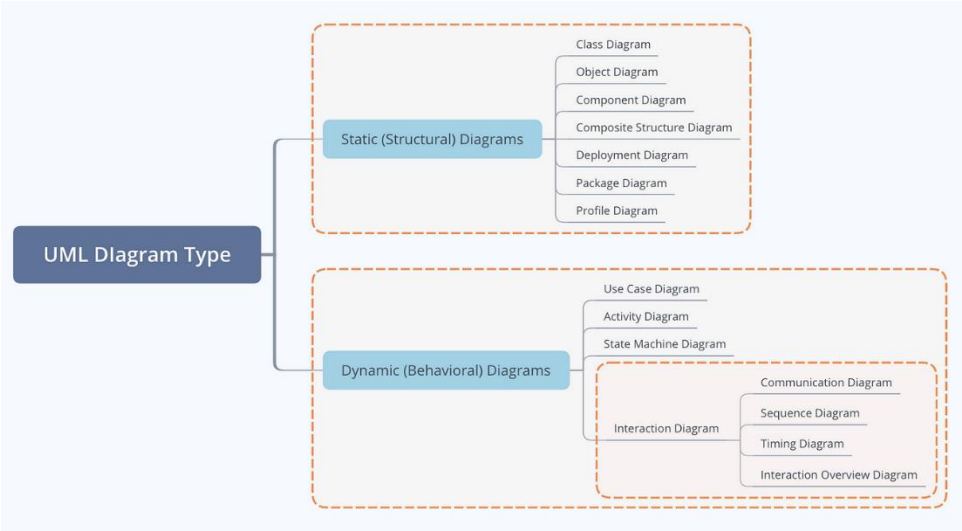

Рисунок 3.3 – Види UML діаграм

Структурні діаграми зображують статичні аспекти системи, які представляють ті частини діаграми, які утворюють основну структуру і тому є стабільними.

Діаграми поведінки показують, як система працює та взаємодіє із зовнішніми об’єктами та користувачами, як вона реагує на введення чи подію та за яких обмежень вона працює.

Діаграми класів є найважливішими діаграмами UML, які використовуються для розробки програмного забезпечення. Діаграма класів описує статичну структуру моделі чи системи у контексті класів об’єктно орієнтовної мови програмування, а також типи об’єктів у системі та різні види зв’язків, які існують між ними. Клас – це структура яка об’єднує в собі дані та методи для роботи із ними. Класи можуть відношення між собою – бути батьківськими, або наслідувати інші класи.

Діаграму класів для модулю захисту інформації зображено на рисунку 3.4.

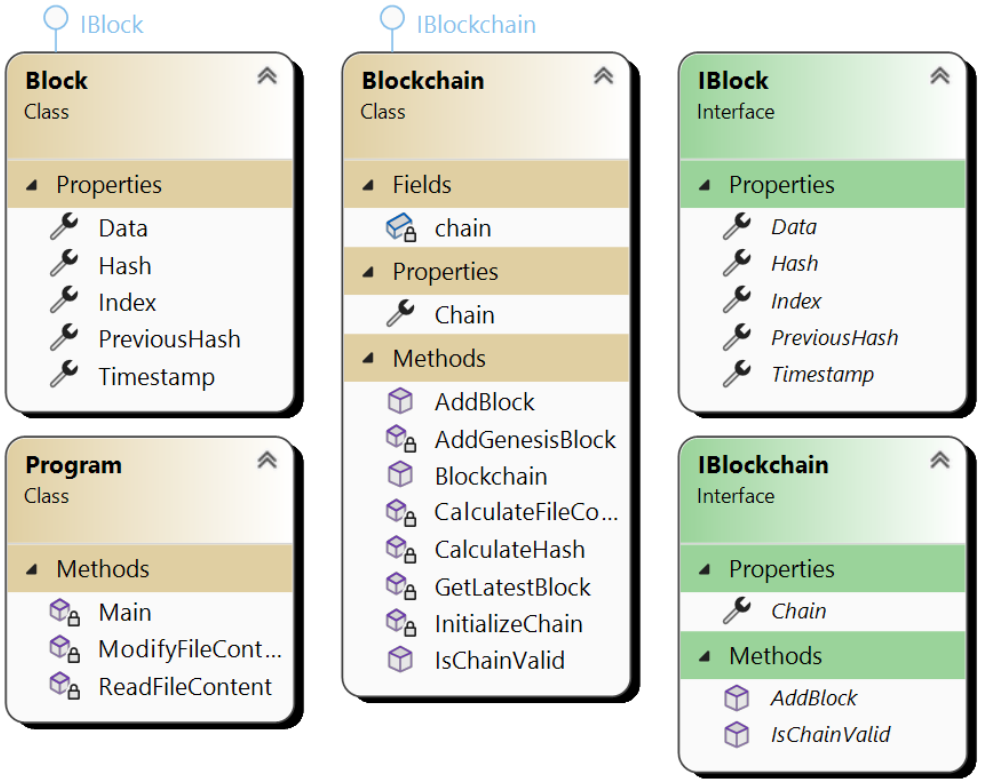

Рисунок 3.4 – Діаграма класів модулю захисту інформації

Із діаграми класів, можемо побачити, основні класи для роботи із захистом інформації, їх методи та властивості. Деякі класи реалізують інтерфейси, що дозволяє створити певний контракт – класи, які реалізовують інтерфейси – зобов'язані імплементувати певні функції. Окрім цього, даний підхід дозволяє у майбутньому замінювати одну реалізацію іншою, без необхідності суттєво змінювати додаток, достатньо імплементувати інтерфейс та використати його нову реалізацію. Наслідування є одним із ключових принципів об'єктно орієнтованих мов та дозволяє також частково позбутися від дублювання коду.

В додатку Б Б.2 зображено діаграму класів для веб серверу. Можемо побачити, що класи контролерів також використовують наслідування – існує базовий клас ApplciationController, який інкапсулює базові методи.

Діаграма компонентів[27] - це деталі реалізації моделі, а діаграми компонентів візуалізують вигляд впровадження системи. Діаграму компонентів зображено на рисунку 3.5.

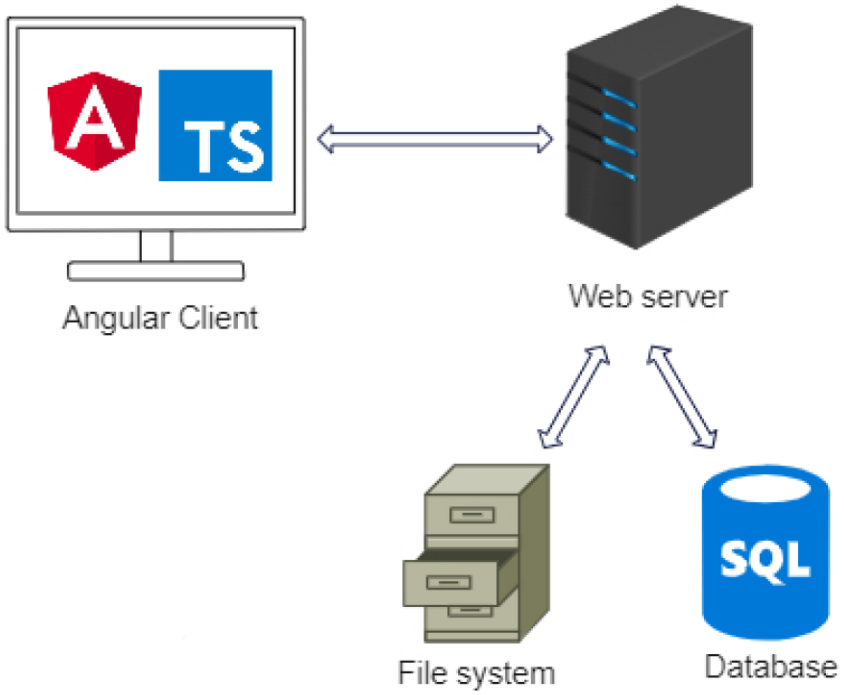

Рисунок 3.5 – Діаграма компонентів системи

Діаграма використання - це моделювання функціональних можливостей, яка дозволяє показати, яким чином ті чи інші учасники взаємодії можуть використовувати систему.

Серед акторів для захищеного сховища даних було виділено користувача, бекенд сервер, фронтенд клієнт, файлову систему та базу даних.

Користувач може автентифікуватися чи зареєструватися у системі – це дозволить отримати доступ до інших функцій, серед яких завантаження, видалення та скачування даних. Важливо, що взаємодія клієнта із системою проходить через фронтенд клієнт, який дозволяє отримувати та відправляти дані на бекенд сервер, який у свою чергу взаємодії з іншими акторами. Фронтенд сервер відповаідає за відправку запитів. Бекенд сервер обробляє запити, виконує роботу із даними, шифруванням даних та автентифікує і авторизує користувача. Файлова система працює із даними. База даних працює із даними та шифруванням даних.

Діаграму використання зображено на рисунку Б.3 в додатку Б.

3.4 Проєктування бази даних

При розробці будь якої програми, важливою частиною бізнес логіки є збереження даних. Це різноманітні дані з доменної області додатку. Для забезпечення автентифікації та авторизації в додатку, за основу було взято структуру бази даних Identity, яка використовується у C#. Схему таблиць бази даних зображено на рисунку Б.4 у додатку Б.

3.5 Висновки

У даному розділі було описано модель зберігання даних. Для забезпечення безпеки конфіденційної інформації, було вирішено скористатися схемою каскадного шифрування з метою захисту від крадіжки або несанкціонованого доступу, а також використовувати технологію блокчейн для запобігання підміні даних у файлі.

Описано алгоритм захищеного збереження даних із використанням технологій блокчейн.

Було обрано архітектурний підхід, проаналізовано переваги та недоліки основних системних архітектур.

Із використанням UML діаграм було спроектовано основні класи системи, показано варіанти використання та послідовність дій у системі. Також графічно було зображено схему бази даних, яка використовується для контролю доступу користувачів до додатку.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАХИЩЕНОГО СХОВИЩА ДАНИХ ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ БЛОКЧЕЙН

4.1 Розробка серверної та клієнтської частин

Розробку серверної частини було почато з створення нового проекту у Visual Studio 2022. Як можемо побачити, хоч додаток і має монолітну архітектуру, проте файли структуровано у папки, відповідно до їх відповідальності, це дозволяю пришвидшити процес розробки та стандартизувати структуру проекту відповідно до загальноприйнятих вимог. Структуру проекту веб серверу зображено на рисунку 4.1.

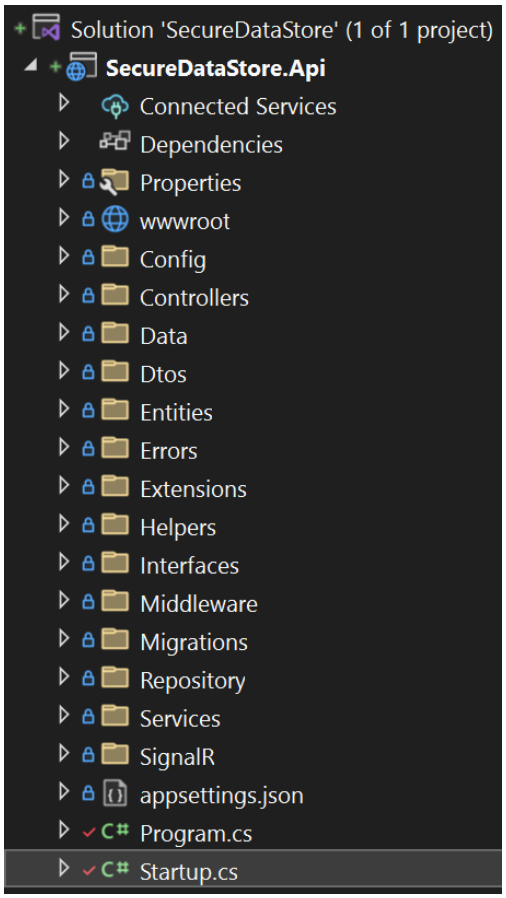

Рисунок 4.1 – Структура проекту веб серверу

Можна виділити такі основні папки: wwwroot містить статичні файли, Controllers – файли контролерів, які інкапсулюють логіку обробки http запитів, Migrations – файли міграцій для entity framework, які відповідають за створення

структури таблиць бази даних, Services – сервіси, так називають класи, які містять бізнес логіку.

На рисунку 4.2 зображено структуру веб клієнту, реалізованого на технології Angular.

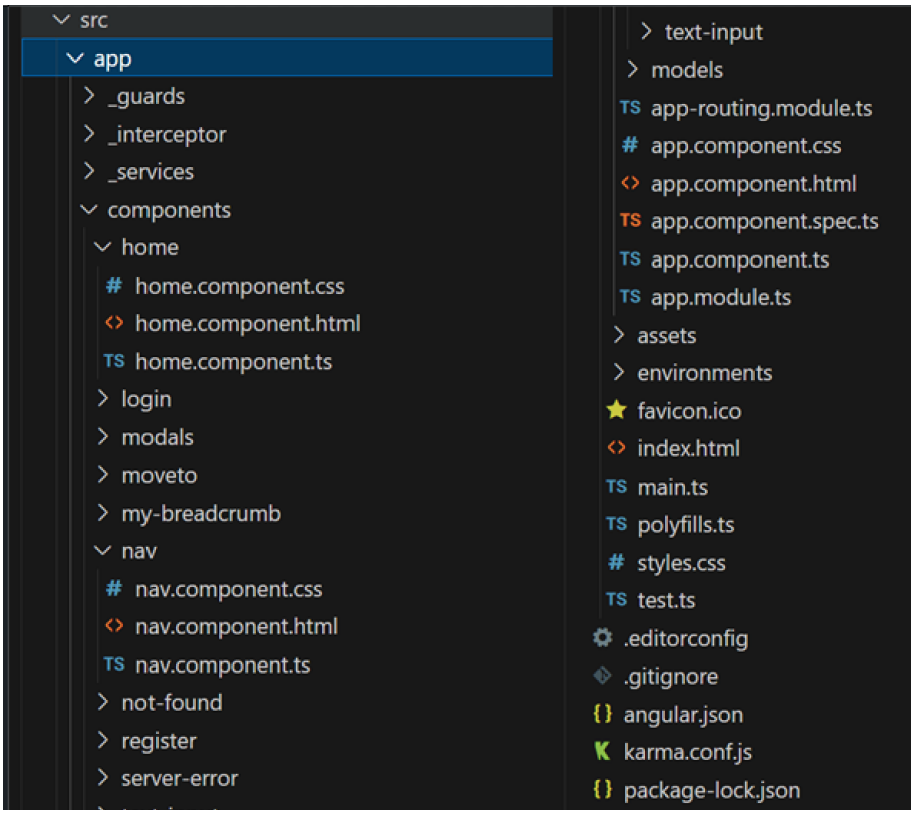

Рисунок 4.2 – Структура клієнтського додатку.

Серед основних частин: папка _services містить сервіси, які інкапсулюють в собі логіку роботи із даними, які отримуються із бекенду, _interceptor – сутності, які задіяні у виконанні http запиту і використовуються для отримання jwt токену та обробки виключень, components – основні будівельні блоки, які містять html, css та typescript і відповідають за відображення контенту, із використанням сервісів для обробки даних перед показом.

Важливою частиною розробки будь якого додатку є налаштування веб серверу, у c# web api за налаштування відповідає клас Startup.

Налаштування веб серверу зображено на рисунку 4.3.

```
2 references | 0 changes | 0 authors, 0 changes
public class Startup
{
    readonly string MyAllowSpecificOrigins = "_myAllowSpecificOrigins";

    0 references | 0 changes | 0 authors, 0 changes
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    3 references | 0 changes | 0 authors, 0 changes
    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    0 references | 0 changes | 0 authors, 0 changes
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllers();
        services.AddApplicationServices(Configuration);//ApplicationServiceExtensions
        //AddCors here with http://localhost:4200 domain of angular
        services.AddCors(options =>
        {
            options.AddPolicy(name: MyAllowSpecificOrigins,
                              builder =>
                              {
                                  builder.WithOrigins("http://localhost:4200", "https://localhost:4200")
                                  .AllowAnyHeader()
                                  .AllowAnyMethod()
                                  .AllowCredentials();
                              });
        });

        services.AddIdentityServices(Configuration);
        services.AddSignalR();
        services.AddSwaggerGen(c =>
        {
            c.SwaggerDoc("v1", new OpenApiInfo { Title = "SecureDataStore", Version = "v1" });
        });
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    0 references | 0 changes | 0 authors, 0 changes
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
            app.UseSwagger();
            app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "SecureDataStore v1"));
        }
        app.UseMiddleware<ExceptionMiddleware>();

        app.UseHttpsRedirection();

        app.UseRouting();
        app.UseCors(MyAllowSpecificOrigins);

        app.UseAuthentication();
        app.UseAuthorization();
```

Рисунок 4.3 – Налаштування веб серверу

Swagger[28] (OpenAPI) - це специфікація, що не залежить від мови програмування, яка надає стандартизований інтерфейс для опису REST API. Це дозволяє як комп’ютерам, так і людям зрозуміти функціонал REST API без прямого доступу до вихідного коду, тобто не обов’язково бути айті спеціалістом і вміти читати код, щоб зрозуміти функції які виконує система. Цей підхід дозволить подати можливості системи у графічній, зрозумілій для всіх формі. Його головні цілі це звести до мінімуму обсяг роботи, необхідний для підключення відокремлених служб та скоротити кількість часу, необхідного для точного документування функцій системи. Специфікація OpenAPI - це документ, який описує можливості API. Документ будується на основі анотацій XML і атрибутах у контролерах і моделях. На

основі цього документу будується відповідна графічна частина яка відображає всі модулі API. Це і є документація і візуалізації функцій системи, яка дозволяє людям без технічної освіти зрозуміти призначення системи, а для розробників надає зручне, задокументоване API і спрощує подальшу інтеграцію і використання API з іншими ресурсами.

Розглянемо задокументовані функції API для захищеного сховища даних із використанням технологій блокчейн. На рисунку 4.4 зображено методи контролеру Account.

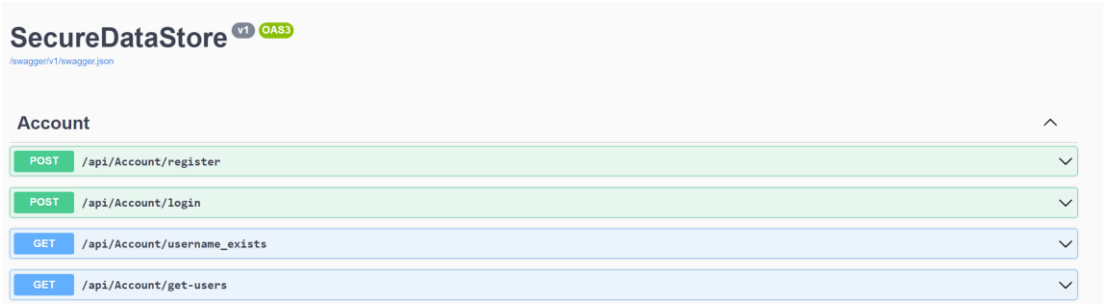

Рисунок 4.4 – Методи котролеру Account

Даний контролер відповідає за управління користувачем. Можемо помітити, що використано GET та POST запити. Серед доступних функцій – створення нового користувача, вхід у систему, перевірка чи ім’я користувача вже використовується і отримання користувачів.

Контролер Data відповідає за обробку запитів, спрямованих на виконання операцій із даними, створення папок, завантажування, скачування, перейменування та інші.

На рисунку 4.5 зображено методи для контролеру Data.

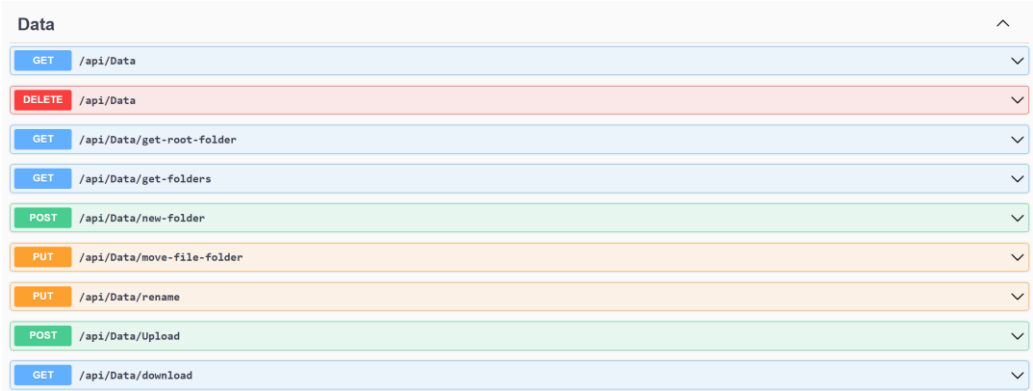

Рисунок 4.5 – Методи котролеру Data

Додаток містить форму авторизації користувача. Є можливість увійти в існуючий аккаунт або створити новий. Форму створення нового користувача в додатку зображено на рисунку 4.6.

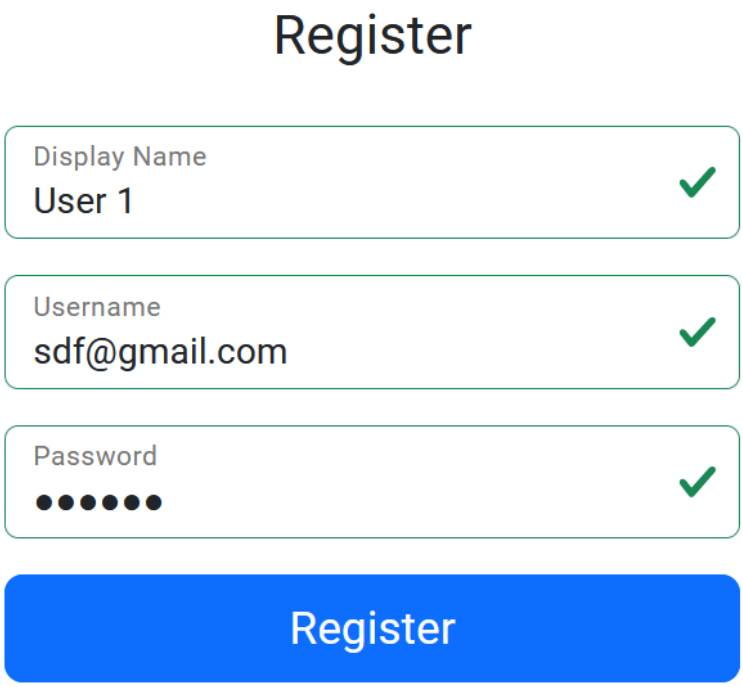

Рисунок 4.6 – Форма створення нового користувача

Після створення нового користувача необхідно автентифікуватися в додатку, для цього слугує форма входу. Рисунок 4.7 відображає форму входу.

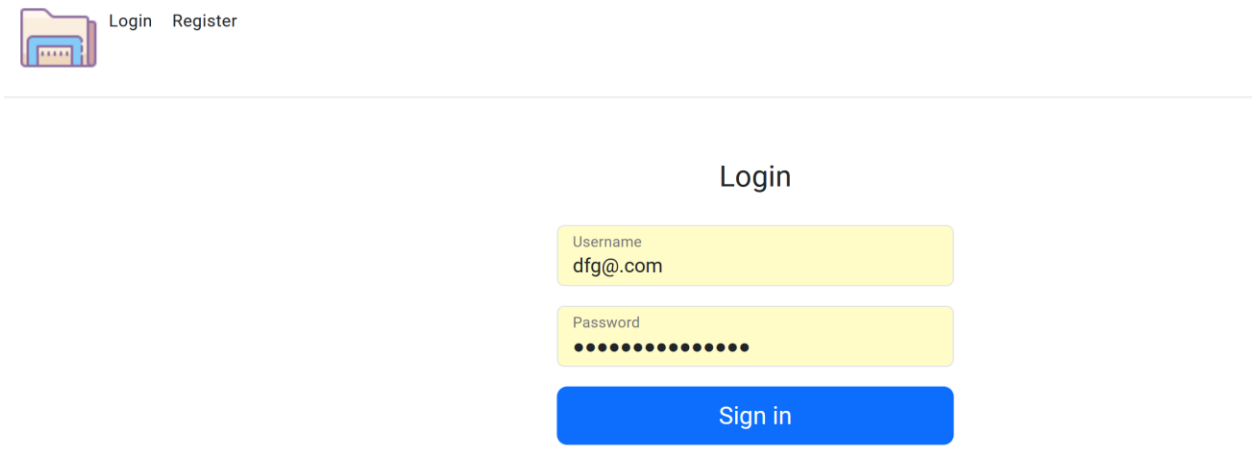

Рисунок 4.7 – Форма входу

Після успішного входу, користувач отримує доступ до функцій додатку і може вільно користуватися ним. На рисунку 4.8 зображено основну сторінку додатку. Користувач може бачити кількість об’єктів у даній директорії, має доступ до створення нових папок та завантаження файлів.

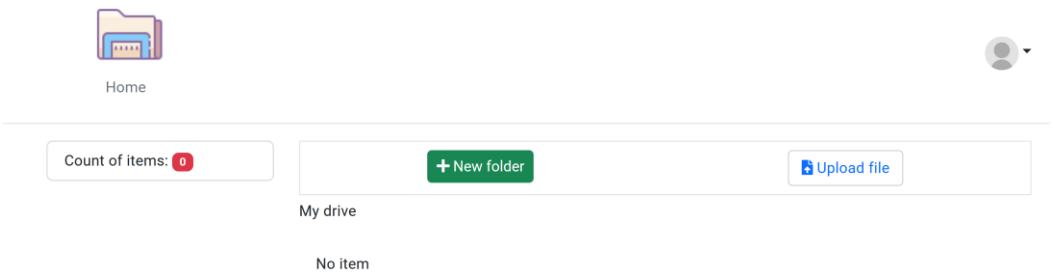

Рисунок 4.8 – Основна сторінка додатку

Процес створення нової папки та завантаження файлу зображено на рисунках 4.9 – 4.11.

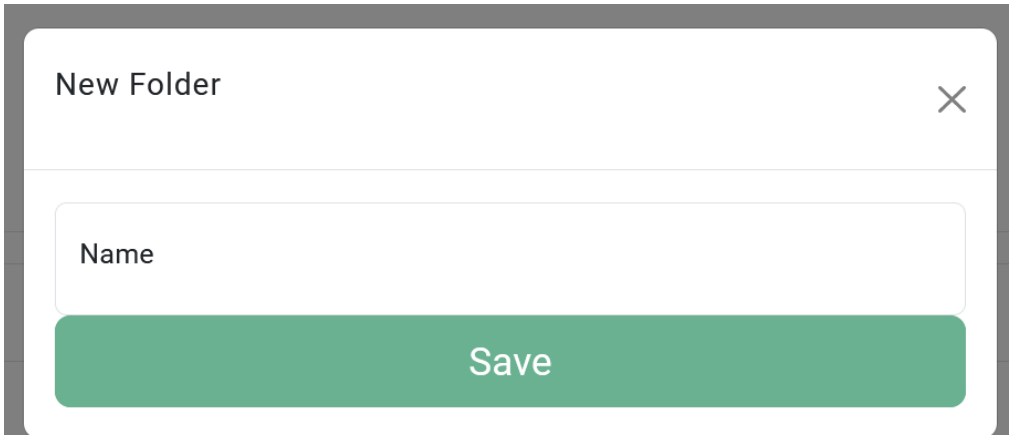

Рисунок 4.9 – Створення нової папки

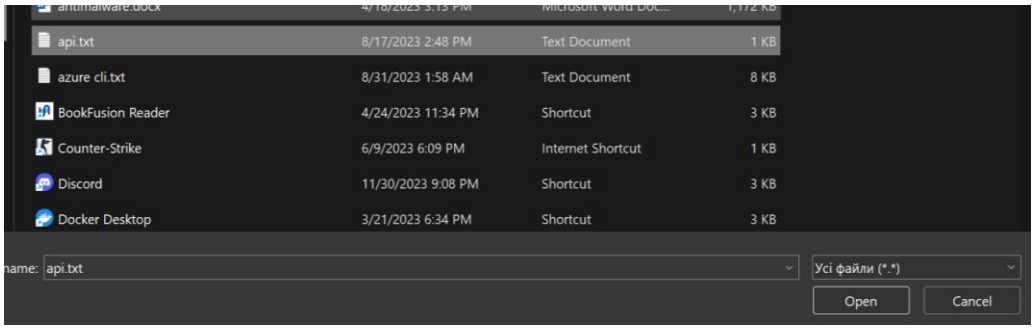

Рисунок 4.10 – Завантаження файлу

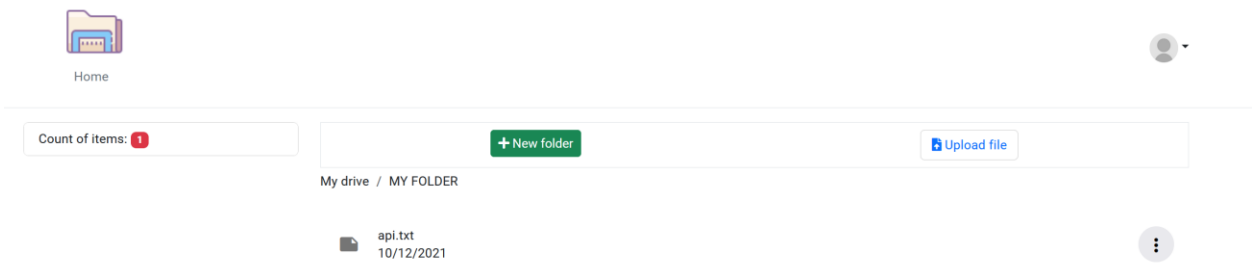

Рисунок 4.11 – Завантажений файл

Після завантаження користувачеві доступні функції для роботи із файлами, які з’являються після кліку на іконку контекстного меню, ці функції зображено на рисунку 4.12.

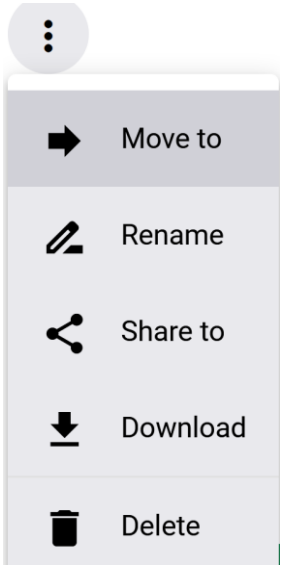

Рисунок 4.12 – Функції для роботи із завантаженим файлом

При кліку на картинку профілю, є можливість вийти із системи. Даний функціонал зображено на рисунку 4.13.

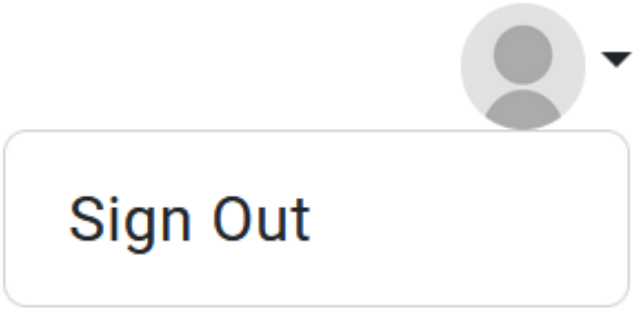

Рисунок 4.13 – Кнопка для виходу з системи

4.2 Створення бази даних

Для створення бази даних було вирішено використати підхід Code First. Entity Framework представив підхід Code-First у Entity Framework 4.1. Code-First в основному корисний у доменно-орієнтованому проектуванні. У підході Code-First програміст зосереджується на домені програми та створює класи для сутності домену, на відмінну від стандартного підходу, коли в першу чергу створюється база даних, а потім створюються класи, які відповідають дизайну бази даних. Рисунок 1.14 ілюструє підхід Code-First.

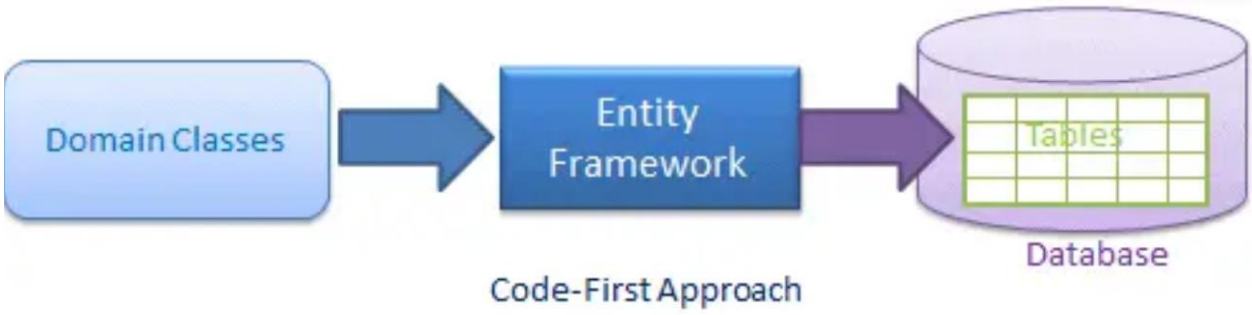

Рисунок 4.14 – Code-First підхід

Як видно із рисунку вище, EF API створить базу даних на основі класів і конфігурації домену програми. Це означає, що потрібно спочатку створити класи, використовуючи мову програмування C# або сумісну із dotnet фреймворком, а потім EF згенерує базу даних на основі існуючого коду. Для того, щоб створити базу даних, після створення всіх базових класів, необхідно відкрити консоль та запустити команду update-database у корені проекту, після чого, базу даних буде створено.

4.3 Розгортання додатку

Розгортання додатку дозволяє зробити додаток доступним для широкого загалу. Проте, зважаючи на те, що додаток слугує для захисту інформації, не кожен схоче довіряти тому чи іншому постачальнику, том убуде описано два варіанти розгортання: локально та на хмарному середовищі Azure.

Для локального розгортання необхідно встановити необхідне програмне забезпечення, а саме: .NET SDK, Node.js. За допомогою команди npm install -g @angular/cli – треба встановити Angular CLI[30].

Після встановлення необхідного програмного забезпечення необхідно відкрити консоль чи командний рядок, перейти до каталогу веб проекту, далі відновити пакети та застосувати міграції за допомогою команд:

dotnet restore

dotnet ef database update

dotnet run

Виконання команд для запуску бекенду зображено на рисунку 4.15.

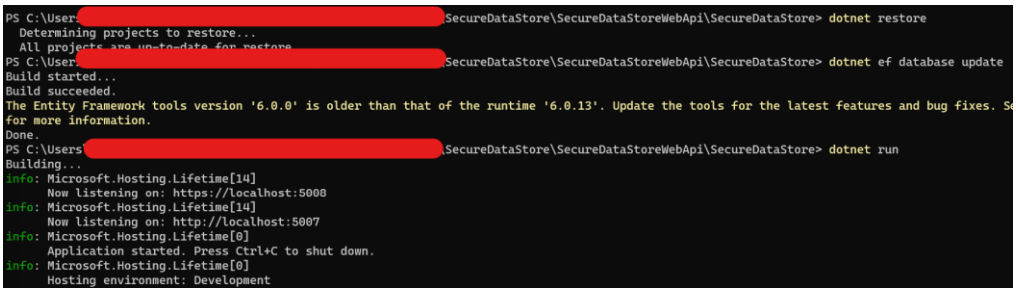

Рисунок 4.15 – Виконання команд для запуску бекенду

Для запуску клієнта Angular необхідно відкрити консоль чи командний рядок та перейти до каталогу проекту клієнта Angular. Після чого встановити пакети npm та запустити додаток командами npm install та npm start.

Виконання команд для запуску Angular клієнту зображено на рисунку 4.16.

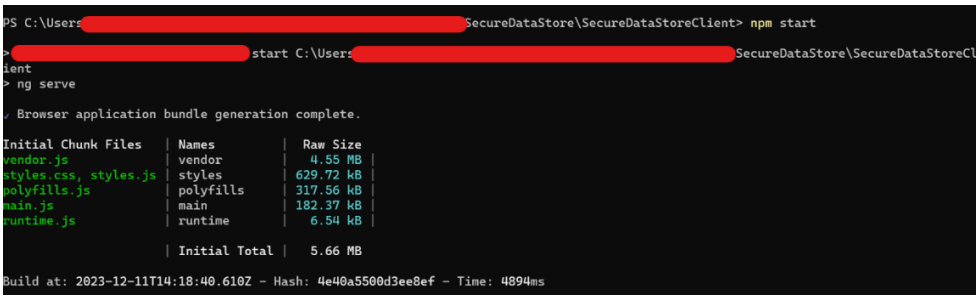

Рисунок 4.16 – Виконання команд для запуску Angular клієнту

Після того як веб-сервіс та клієнт Angular запущені, можна отримати доступ до додатку у веб-браузері, перейшовши за адресою https://localhost:4200.

Для публікації проекту необхідно відкрити проект у середовищі розробки Visual studio та натиснути правою кнопкою на назву проекту, після чого із контекстного меню обрати команду “Publish”, як зображено на рисунку 4.17.

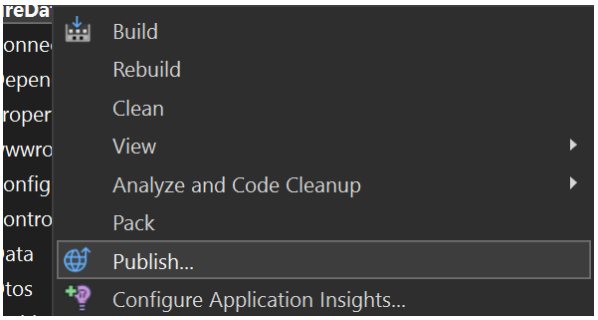

Рисунок 4.17 – Контекстне меню Visual studio із кнопкою “Publish”

Наступним кроком необхідно обрати середовище для публікації додатку, серед списку доступних оберемо Azure, як зображено на рисунку 4.18.

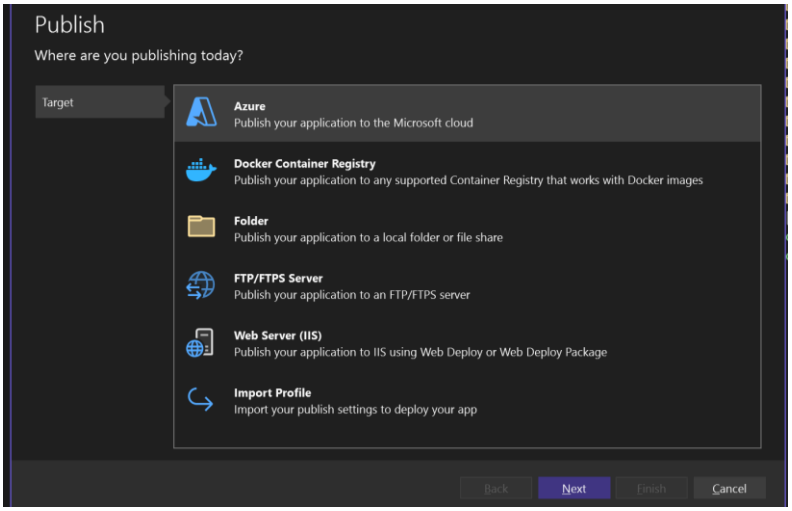

Рисунок 4.18 – Вибір Azure як середовища для розгортання додатку

Після того, як було обрано Azure, необхідно обрати ажур акаунт, даний процес зображено на рисунку 4.19.

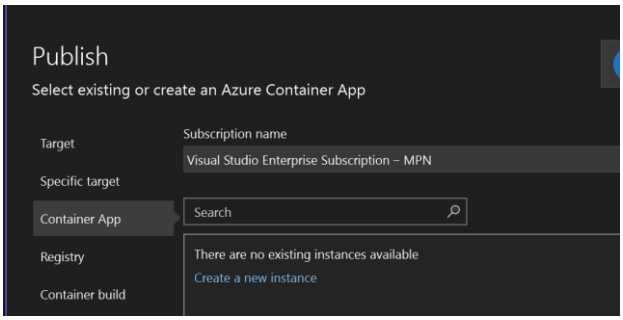

Рисунок 4.19 – Вибір ажур акаунту

Далі необхідно використати вже існуючі або створити нові ресурси які будуть використовуватися для розгортання проекту, можна обрати або створити назву API, конкретну підписку на ажур, групу ресурсів та сервіс управління API. Створення ресурсів для API зображено на рисунку 4.20.

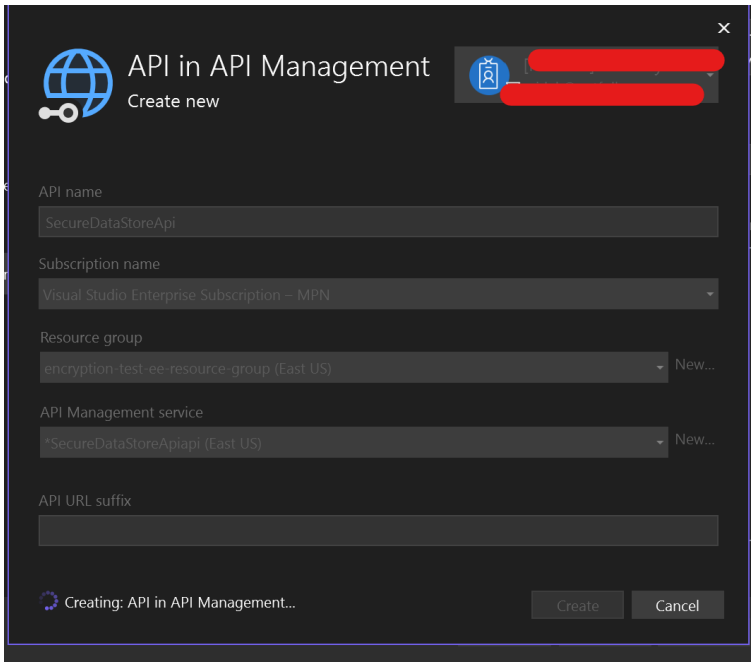

Рисунок 4.20 – Створення ресурсів API

Створення ресурсів може зайняти певний час, необхідно дочекатися завершення процесу, після чого з’явиться вікно із повідомленням про завершення публікації. Можна використовувати проект. Розгортання клієнтської частини відбувається аналогічним чином.

4.4 Тестування додатку

Розробка програмного забезпечення великою мірою залежить від процесу тестування, який відіграє важливу роль. Виявлення помилок під час розробки є набагато більш вигідним, ніж у випадку, коли проблема стикається користувачем і спричиняє появу помилок та проблем.

Тестування[31] може бути класифіковане на наступні види:

- Тестування Black Box включає в себе перевірку системи, де код і функціонал не є відомими особі, яка проводить тестування;
- End-to-End (E2E) - це техніка, яка перевіряє робочий процес програми від початку до кінця, щоб гарантувати правильну роботу всіх компонентів;
- Функціональне тестування – це еревірка програми, веб-сайту або системи для впевненості, що вони виконують необхідні функції;
- Інтеграційне тестування гарантує, що інтегрована система відповідає вимогам у всій своїй комплексності;
- Тестування навантаження визначає, як програма веде себе при одночасних діях великої кількості користувачів;
- Нефункціональне тестування – це перевірка готовності системи за нефункціональними параметрами, такими як продуктивність, доступність та UX;
- Тестування продуктивності – це оцінка швидкості, стабільності, надійності, масштабованості та використання ресурсів під навантаженням;
- Юніт тестування – це перевірка окремих фрагментів коду для впевненості, що вони працюють правильно, що сприяє ефективності тестування та скороченню кількості зайвих тестів.

Для тестування додатку було вирішено використати юніт тести, підхід мануального тестування, а також перевірити швидкодію розробленого алгоритму. Юніт тести ідеально підходять для тестування окремих фрагментів програми, можна впевнитися, що модулі працюють вірно. Юніт тести було застосовано для перевірки коректності роботи програмної частини, а саме – модулю шифрування. Це рішення дозволить із більшою ймовірністю, ніж при ручному тестуванні впевнитися в коректності роботи даного модуля.

Запуск юніт тестів зображено на рисунку 4.21.

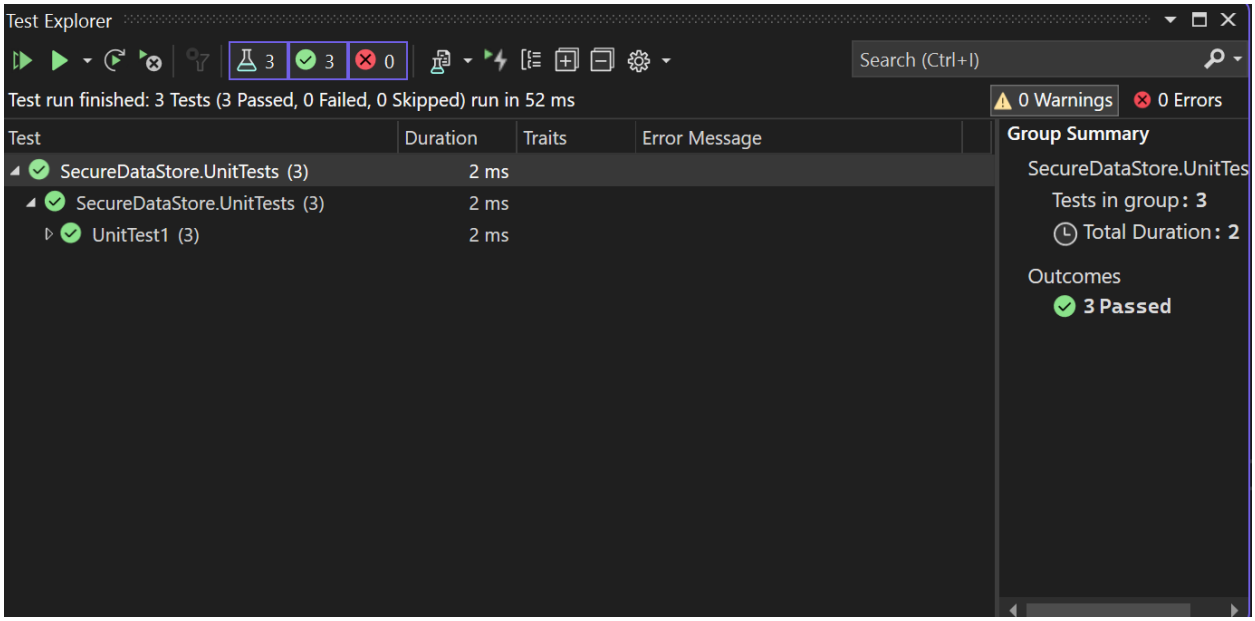

Рисунок 4.21 – Запуск юніт тестів.

Для перевірки користувацького інтерфейсу було вирішено використати мануальне тестування. Звісно, даний функціонал можна протестувати із використанням інших підходів, проте використання автоматизованих, наприклад юніт тестів, вимагає використання сторонніх бібліотек та витрат часу на написання логіки тестування. Даний підхід виправдовує себе у зрілому продукті, який змінюється не часто, або несуттєво, зміни в якому в основному пов’язані зі змінами в алгоритмах чи функціях при обробці інформації. Адже, після найменшої зміни інтерфейсу, необхідно переписувати автоматизовані тести. Отже, зважаючи на вищеперераховані фактори та невеликий об’єм сторінок та функціоналу клієнтського додатку, для тестування функціоналу було проведено мануальне тестування.

Для тестування форми вводу логіну, не було введено жодного значення, як видно з рисунку 4.22, виводиться валідаційне повідомлення про помилку.

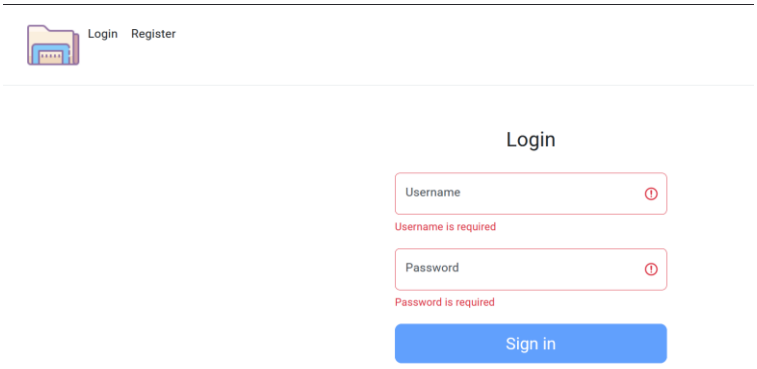

Рисунок 4.22 – Тестування полів введення форми логіну

Тест полів введення форми реєстрації зображено у додатку В на рисунку В.1. Для тестування форми реєстрації було заповнено два із трьох полів, як видно з рисунку 4.23, заповнені поля позначаються зеленим кольором, а незаповнене – червоним, також виводиться валідаційне повідомлення про необхідну довжину.

Рисунок 4.23 – Тестування частини полів введення форми реєстрації

Тестування при коректному введенні даних у форму реєстрації зображено на рисунку 4.24.

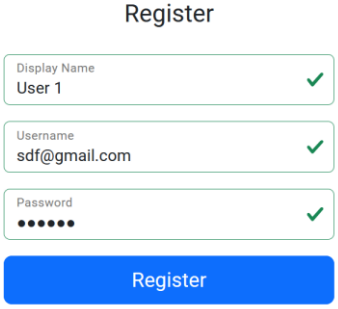

Рисунок 4.24 – Коректне введення даних форми реєстрації

Після успішної реєстрації, користувач може увійти в систему та отримати доступ до функціоналу системи. Введення даних, попередньо створеного користувача зображено на рисунку 4.25.

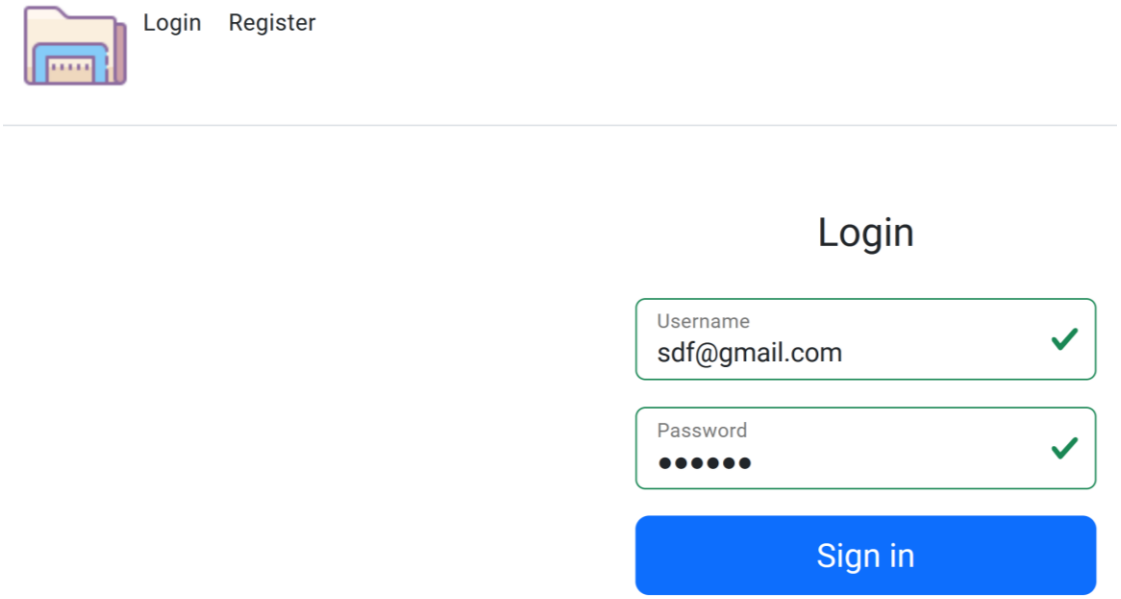

Рисунок 4.25 – Введення коректних даних у форму логіну

Домашній екран додатку після входу, зображено на рисунку 4.26. Проте, при введенні невірних даних, користувач побачить повідомлення про помилку. Рисунок В.2 у додатку В ілюструє помилку при спробі входу в систему із невірними даними.

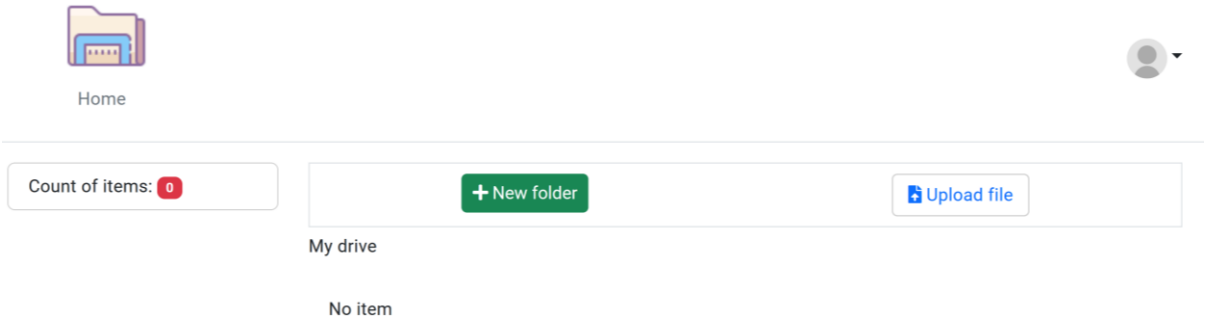

Рисунок 4.26 – Введення коректних даних у форму логіну

При спробі створити папку із порожнім полем назви, користувач отримає помилку, як зображено на рисунку 4.27.

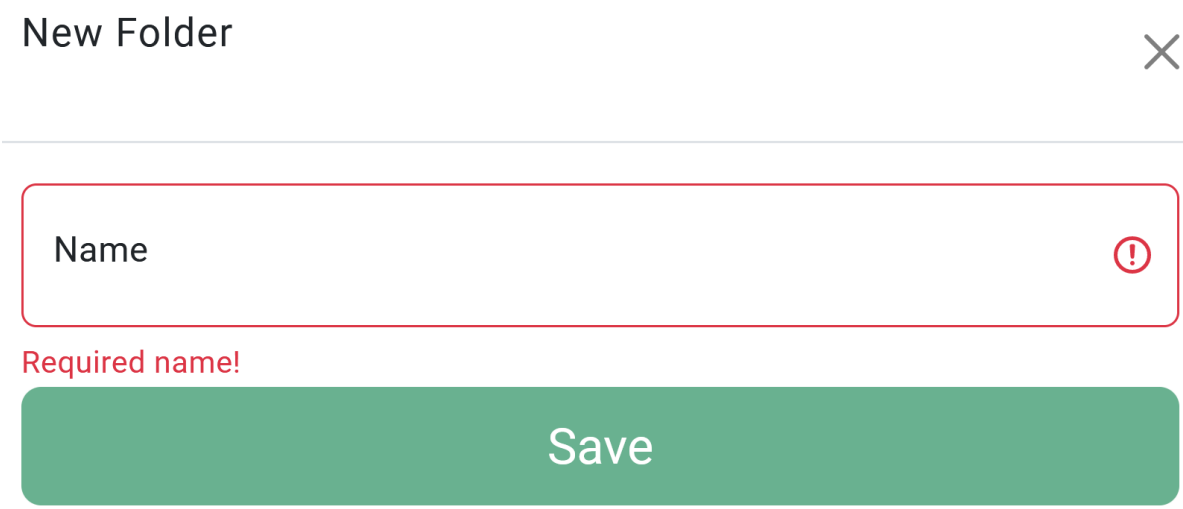

Рисунок 4.27 – Помилка при спробі створити папку із порожнім полем введення назви

Для тестування функціоналу забезпечення цілісності та шифрування було проведено тест із завантаженням файлу, спробою відкрити його без використання додатку та змінити внутрішній зміст файлу.

Рисунок 4.28 відображає вміст тестового файлу, який буде завантажено в систему.

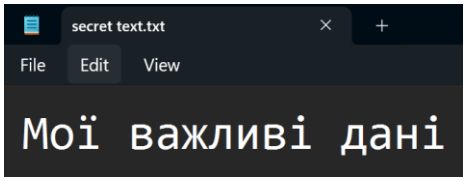

Рисунок 4.28 – Вміст тестового файлу

Завантажений файл зображено на рисунку 4.29.

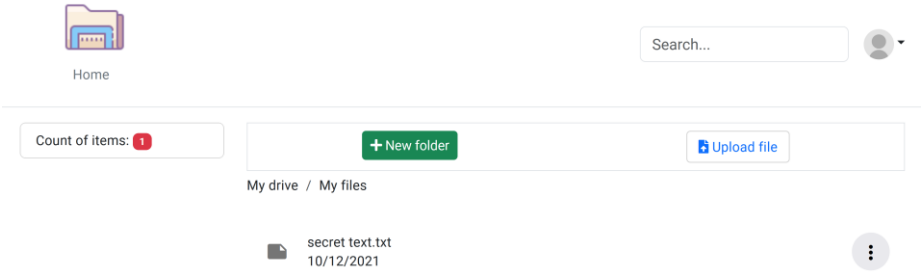

Рисунок 4.29 – Завантажений тестовий файл

Вміст файлу після завантаження та шифрування зображено на рисунку 4.30.

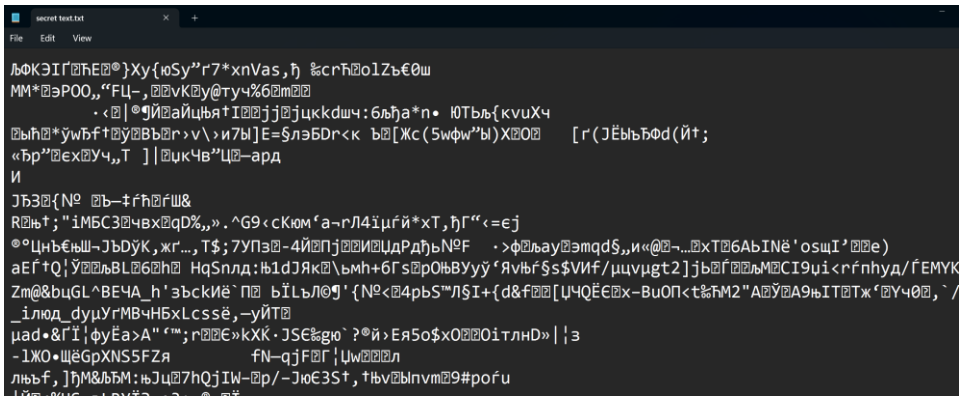

Рисунок 4.30 – Вміст тестового файлу після завантаження у додаток та шифрування

Якщо відкрити зашифрований файл у сторонньому редакторі і змінити його вміст – хеш зміниться, при перевірці документу це буде помічено системою. Файл виділиться червоним кольором, з’явиться повідомлення про те, що вміст файлу було скомпрометовано.

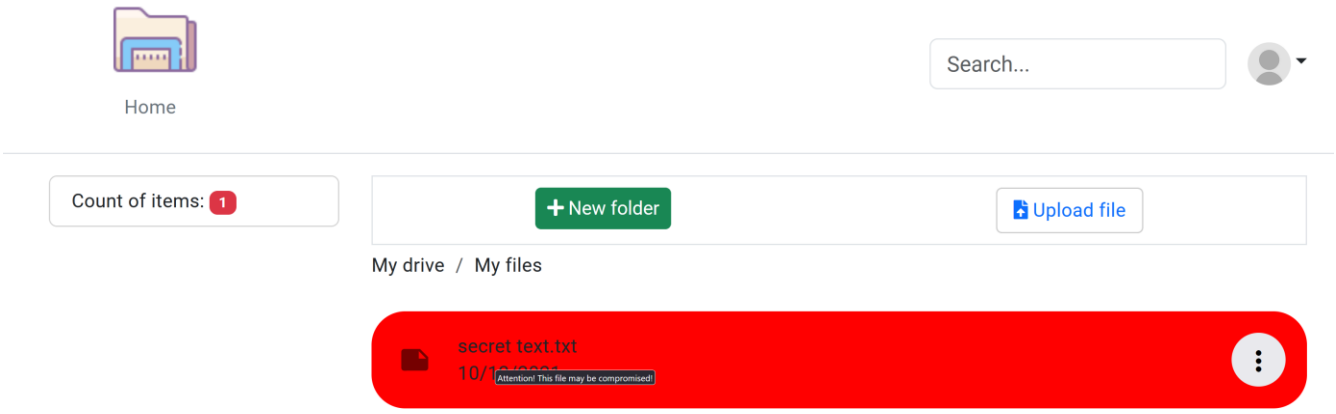

Рисунок 4.31 – Вигляд файлу у додатку після спроби редагувати файл за межами додатку

Результати перевірки швидкодії додатку при шифруванні файлів Винесено у таблиці В.1 і В.2 у додатку В та рисунки Б.5 і В.6 у додатку Б.

4.5 Висновки

В даному розділі було розглянуто процес розробки серверної частини у середовищі написання коду Visual Studio 2022, наведено рисунку які демонструють створення класів, процес налаштування веб серверу та створення бібліотеки класів. Для створення клієнтської частини було використано технологію ASP.NET MVC. Було описано процес створення бази даних із використанням підходу Code First та міграцій.

Було продемонстровано процес розгортання рішення серверах ажур та локально.

Для забезпечення якості додатку та відсутності багів було проведено тестування модулю шифрування із використанням юніт тестів. Здійснено тестування основного функціоналу клієнтської частини та елементів користувацького інтерфейсу. Для модулю шифрування проведено окреме тестування на час виконання операцій. Отримані результати внесено у таблицю у додатку В.

5. ЕКОНОМІЧНИЙ РОЗДІЛ

5.1 Технологічний аудит розробленого захищеного сховища даних із використанням технологій блокчейн

В наш час гостро постає питання конфіденційності та безпеки інформації. Часи змінилися і відбувається глобальна діджиталізація. Сотні гігабайтів фото, відео та документів містять наші комп'ютери. У певних випадках, це не просто персональні фото, а конфіденційна інформація і її власник не бажає ділитися нею. Чи безпечно довіряти свої дані корпораціям чи зберігати їх у месенджерах – вибір кожного. Проте із кожним днем все більше людей переймаються своєю конфіденційністю та борються за право на приватність.

Метою даної магістерської кваліфікаційної роботи була розробка сховища даних, яке зменшить ймовірність компрометації файлів та втрати приватних даних, забезпечить надійне збереження файлів користувача, щоб навіть у разі втрати даних – зловмисники не змогли скористатися вкраденим.

Для реалізації мети було проведено порівняльний аналіз існуючих систем, описано методи і моделі збереження даних, проаналізовано програмну реалізацію захищеного сховища даних, архітектуру системи та бази даних, і з використанням мов програмування C# та Angular створено програмне забезпечення та проведено тестування готової системи.

Як результат отримано гібридний алгоритм шифрування, який поєднує у собі каскадне шифрування та технологію блокчейн і забезпечує надійний захист та зменшує ймовірність компрометації файлів, а також програму яка використовує даний алгоритм для збереження файлів.

Для встановлення комерційного потенціалу розробленого захищеного сховища даних із використанням технологій блокчейн було проведено його технологічний аудит і запрошено 3-х відомих і знаних в Україні експертів: доцента Гармаша В. В., доцента Кабачія В. В. та доцента Овчинникова К. В. Встановлення комерційного потенціалу розробленого захищеного сховища даних із використанням технологій блокчейн було здійснено за критеріями, наведеними в таблиці 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання технічного рівня та комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Запрошені експерти оцінили розроблене захищене сховища даних із використанням технологій блокчейн так, як показано в табл. 5.2:

Таблиця 5.2 – Результати технологічного аудиту розробленого захищеного сховища даних із використанням технологій блокчейн (за шкалою оцінювання «0»-«1»-«2»-«3»-«4»)

Критерії	Прізвище, ініціали експертів		
	Квєтний Р.Н.	Гармаш В.В.	Барабан М.В.
	Бали, що їх виставили експерти:		
1	3	3	3
2	4	3	4
3	3	3	3
4	3	4	3
5	4	3	4
6	4	4	3
7	4	3	4
8	3	4	3
9	4	4	3
10	4	3	3
11	4	3	3
12	4	4	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 41$	$СБ_3 = 40$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{44+40+41}{3} = \frac{125}{3} = 41,67$		

Встановлення комерційного потенціалу розробленого захищеного сховища даних із використанням технологій блокчейн було зроблено на основі рекомендацій, наведених в таблиці 5.3 [29].

Таблиця 5.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 41,67 балів, то це свідчить, що розроблене захищене сховище даних із використанням технологій блокчейн має рівень комерційного потенціалу, який вважається

«високим». Це пояснюється тим, що розроблене сховище пропонує надійний алгоритм шифрування та забезпечує незмінність даних.

5.2 Розрахунок витрат на розробку захищеного сховища даних із використанням технологій блокчейн

При розробці захищеного сховища даних із використанням технологій блокчейн були зроблені такі витрати:

А). Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн}, \qquad (5.1)$$

де М – місячний посадовий оклад розробника, грн; приймемо, що

М = (6700…27200) грн/місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 22 дні;

t – число днів роботи розробників.

Зроблені розрахунки зведемо до таблиці 5.4:

Таблиця 5.4 – Основна заробітна плата розробників

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	20000	1000	20 годин	≈ 3333
2. Магістрант-студент-виконавець	2000 (беремо 6700)	319,04	70	≈ 22333
3. Консультант з економічної частини	17000	809,53	1,5 години	≈ 203 (при 6-годинному робочому дні)
Загалом				$З_о$ = 25 869 грн

Б). Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о . \qquad (5.2)$$

Приймемо, що α = 0,11. Тоді для нашого випадку отримаємо:

$$З_д = 0{,}11 \times 25869 = 2845{,}59 \approx 2846 \text{ грн}.$$

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (5.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. β = 22%. Тоді:

$$НЗН_{зп} = (25869 + 2846) \times 0{,}22 = 6317{,}3 \approx 6317 \text{ грн.}$$

Г). Амортизація основних засобів А, які використовувались під час виконання цієї роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн}, \quad (5.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань; $Н_а$ = (2,5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 5.5.

Таблиця 5.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання тощо	48000	25	3,1 (при 85% використанні	2635
2. Приміщення університету, кафедри	22000	3,5	3,1 при 90% використанні	≈ 179
Всього				А = 2814 грн

Д). Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_і \cdot Ц_і \cdot К_і - \sum_{1}^{n} В_і \cdot Ц_в \text{ грн.}, \quad (5.5)$$

де $Н_і$ – витрати матеріалу *і*-го найменування, кг; $Ц_і$ – вартість матеріалу *і*-го найменування; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); $В_і$ – маса

відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ грн}, \quad (5.6)$$

де $Н_i$ – кількість комплектуючих *і*-го виду, шт.; $Ц_i$ – ціна комплектуючих *і*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання роботи загальні витрати на матеріали та комплектуючі склали приблизно 1200 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \quad (5.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2023 р. В ≈ 4,5 грн/кВт;

П – установлена потужність обладнання, кВт; П = 0,85 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 260 годин;

$К_п$ – коефіцієнт використання потужності; $К_п$ < 1 = 0,8.

$К_д$ – коефіцієнт корисної дії, $К_д$ = 0,7.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot Ф \cdot К \cdot _п}{К_д}$$

$$Be = \ 4{,}5 \cdot 0{,}85 \cdot 260 \cdot 0{,}8)/0{,}7 = 1136{,}57 \approx 1137 \text{ грн.}$$

Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0{,}5....3) \times З_о. \quad (5.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 1{,}6 \times 25\ 869 = 41390 \text{ грн.}$$

К). Сума всіх попередніх статей витрат складає витрати на виконання роботи безпосередньо розробником-магістрантом – В.

$$В = 25869 + 2846 + 6317 + 2814 + 1200 + 1137 + 41390 = 81573 \text{ грн.}$$

Л). Загальні витрати на розробленого захищеного сховища даних із використанням технологій блокчейн $В_{заг}$ становлять:

$$В_{заг} = \frac{В}{\beta}, \quad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0{,}88$[29], оскільки робота практично завершена.

Тоді: $Взаг = 81573/0{,}88 = 92696{,}59$ грн або приблизно 92697.

Тобто прогнозовані загальні витрати на розробку захищеного сховища даних із використанням технологій блокчейн становлять приблизно 92 тисячі шістсот дев'яносто сім грн.

5.3 Розрахунок економічного ефекту від можливої комерціалізації даної розробки

Економічний ефект від можливої комерціалізації розробленого захищеного сховища даних із використанням технологій блокчейн полягає у кращих алгоритмах, які використовуються для захисту даних та наявності зручного користувацького інтерфейсу та швидкодії системи.

Попит на даний продукт може виникати у дрібних компаній, які працюють із даними, а також в окремих людей для персонального використання.

Аналіз ринку також показав, що потенційна кількість замовників аналогічних систем становить сьогодні приблизно 10-20 осіб і буде постійно зростати. Для розрахунку було взято середнє значення – 15 осіб. Тобто, якщо дана розробка буде впроваджена з 1 січня 2024 року, то її результати будуть виявлятися протягом 2024-го, 2025-го та 2026-го років.

Прогноз зростання попиту на нашу розробку складає по роках:

а) 2024 р. – + 5 шт. до базового року;

б) 2025 р. – + 10 шт. до базового року;

в) 2026 р. – + 25 шт. до базового року.

У 2022 році подібна за функціями розробка коштувала на ринку приблизно 100 тисяч грн. Тоді дану, значно кращу за своїм функціоналом розробку, можна буде продавати на ринку дорожче, в середньому за 120 тисяч грн або на 5 тисяч грн дорожче.

Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від виведення даної розробки на ринок, становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (5.10)$$

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів даної розробки у цьому році. Для нашого випадку це є збільшення ціни реалізації даної розробки $\Delta Ц_o$ = 120 – 100 = 20 тисяч грн;

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 15 шт.;

ΔN – покращення основного кількісного показника від впровадження результатів розробки.

Таке покращення становитиме по роках, відповідно: у 2024 році – + 5 шт., у 2025 році + 10 шт., та у 2026 році + 25 шт.;

$Ц_o$ – основний якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, грн; $Ц_o$= 120 тисяч грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 3;

λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = (0,2...0,5); візьмемо ρ = 0,5;

υ – ставка податку на прибуток. У 2023-26 роках υ = 19% .

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливого впровадження і комерціалізації даної розробки (2024 р.) становитиме:

$$\Delta П_1 = [10 \cdot 15 + 110 \cdot 5] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - 19/100) \approx 236 \text{ тис. грн,}$$

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливого впровадження і комерціалізації даної розробки протягом другого (2025) року складе:

$$\Delta П_2 = [10 \cdot 15 + 110 \cdot 10] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - 19/100) \approx 421 \text{ тис. грн,}$$

що

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливого впровадження та комерціалізації даної розробки протягом третього (2026) року складе:

$$\Delta П_3 = [10 \cdot 15 + 110 \cdot 25] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - 19/100) \approx 978 \text{ тис. грн.}$$

Приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації даної розробки для інвестора становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де

τ – ставка дисконтування. Приймемо τ = 0,10 (10%);

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від впровадження та комерціалізації даної розробки, складе:

$$ПП = 236 / (1 + 0{,}1)\ \text{^}2 + 421/(1+0{,}1)\text{^}3 + 978/(1+0{,}1)\text{^}4 = 1179 \text{ тисяч грн.}$$

Теперішня вартість інвестицій PV, що повинні бути вкладені у реалізацію розробки: $PV = (1{,}0 \ldots 5{,}0) \times В_{заг}$.

Для нашого випадку $PV = (1{,}0 \ldots 5{,}0) \times 92 = 3{,}0 \times 92 = 276$ тисяч грн.

Абсолютний ефект від можливих вкладених в реалізацію розробки інвестицій $Е_{абс}$.

$$Е_{абс} = ПП – PV, \quad (5.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для потенційного інвестора від можливої комерціалізації даної розробки, грн;

PV – теперішня вартість інвестицій PV = 276 тисяч грн.

Абсолютний ефект від можливого впровадження даної розробки складе:

$$Е_{абс} = 1179 – 276 = 903 \text{ тисяч грн.}$$

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (5.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 903 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 276 тис. грн;

$Т_ж$ – життєвий цикл розробки, роки.

$Т_ж$ = 4 років (2023-й, 2024-й, 2025-й, 2026-й роки)

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{903}{276}} - 1 = \sqrt[4]{1 + 3{,}2717} - 1 = \sqrt[4]{4{,}2717} - 1 = 1{,}437 - 1 = 0{,}437 = 43{,}7\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією даної розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \quad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022-2023 роках в Україні d = (0,10...0,12);

f – показник, що характеризує ризикованість вкладень;

f = (0,1...0,50). Приймемо f = 0,30.

Для нашого випадку отримаємо:

$$\tau_{мін} = 0{,}12 + 0{,}30 = 0{,}42 \text{ або } \tau_{мін} = 42\%.$$

Оскільки величина $E_в = 43,7\% > \tau_{мін} = 42\%$, то потенційний інвестор у принципі може бути зацікавлений у фінансуванні та комерціалізації розробленого захищеного сховища даних із використанням технологій блокчейн.

Розрахунок терміну окупності коштів, вкладених у можливу комерціалізацію розробленого захищеного сховища даних із використанням технологій блокчейн.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}. \qquad (5.15)$$

Для даного випадку термін окупності $T_{ок}$ коштів становитиме:

$$T_{ок} = \frac{1}{0,437} = 2,28 \text{ років} < 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленого захищеного сховища даних із використанням технологій блокчейн.

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених потенційних інвестицій від рівня інфляції в країні.

Якщо рівень інфляції в країні зросте до 20%, то:

$$ПП = \frac{236}{(1+0,2)^2} + \frac{421}{(1+0,2)^3} + \frac{978}{(1+0,2)^4} \approx 164 + 244 + 472 = 880 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження даної розробки за три роки складе:

$$E_{абс} = 880 - 276 = 604 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс} = 604$ тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 276 тисяч грн.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{604}{276}} - 1 = \sqrt[4]{1 + 2,1884} - 1 = \sqrt[4]{3,1884} - 1 = 1,33 - 1 = 0,33 = 33\%.$$

Оскільки величина $E_в$ = 33% $< \tau_{мін}$ = 42%, то потенційний інвестор може не бути зацікавлений у фінансуванні та комерціалізації розробленого захищеного сховища даних із використанням технологій блокчейн.

Якщо рівень інфляції в країні зросте до 30%, то:

$$ПП = \frac{236}{(1+0,3)^2} + \frac{421}{(1+0,3)^3} + \frac{978}{(1+0,3)^4} \approx 140 + 192 + 342 = 674 \text{ тисяч грн.}$$

Тоді абсолютний ефект від можливого впровадження даної розробки за три роки складе:

$$E_{абс} = 674 - 276 = 398 \text{ тисяч грн.}$$

Внутрішня дохідність $E_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1,$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 398 тисяч грн;

PV –теперішня вартість початкових інвестицій PV = 276 тисяч грн.

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{398}{276}} - 1 = \sqrt[4]{1 + 1,44} - 1 = \sqrt[4]{2,44} - 1 = 1,25 - 1 = 0,25 = 25\%.$$

Оскільки величина $E_в$ = 25% $< \tau_{мін}$ = 42%, то потенційний інвестор може бути і незацікавлений у фінансуванні та комерціалізації даної розробки.

Зроблені розрахунки у вигляді графіків наведено на рис. 5.1.

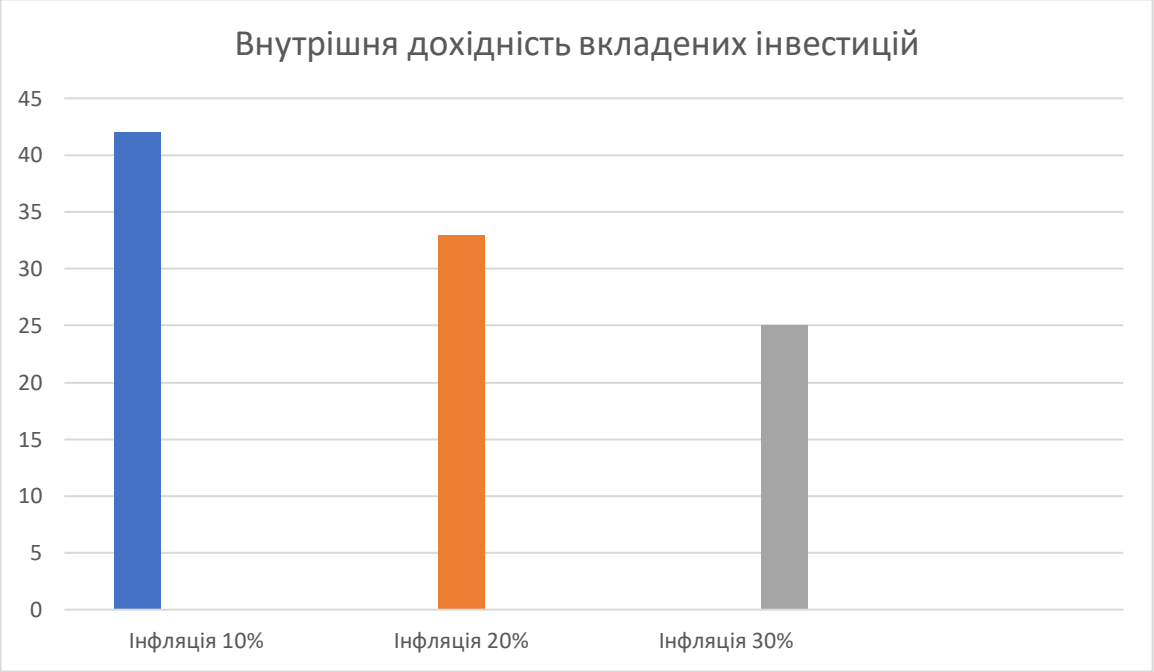

Рисунок 5.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій, вкладених у комерціалізацію даної розробки, від рівня інфляції в країні (10%, 20% і 30%)

Аналіз діаграм на рис 5.1 показує, що при рівні інфляції в 10% величина внутрішньої дохідності інвестицій становить $E_в = 42\%$, що вище порогового значення $\tau_{мін} = 42\%$ і тому комерціалізація даної розробки може бути доцільною. При рівні інфляції в 20% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію даної розробки, становить $E_в = 33\% < 42\%$, а при рівні інфляції в 30% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію даної розробки, становить $E_в = 25\% < 42\%$, тобто нижче порогового рівня $\tau_{мін} = 42\%$, що може викликати у потенційного інвестора сумніви у доцільності комерціалізації даної розробки.

Тому у двох останніх випадках остаточне рішення з питання комерціалізації даної розробки потребує проведення додаткових розрахунків (можливо – зниження рівня ризикованості вкладень, збільшення попиту на розробку, збільшення ціни реалізації розробки тощо).

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю:

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	Не більше 100 тисяч грн	92 тисячі грн	Досягнуто
2. Абсолютний ефект від впровадження розробки, тисяч грн	Не менше 900 тисяч грн	903 тисяч грн (при 10%-інфляції)	Виконано
3. Внутрішня дохідність інвестицій, %	не менше 42%	42% (при 10%-інфляції)	Досягнуто
4. Термін окупності інвестицій, роки	до 3-ти років	2,81 років	Виконано

Таким чином, основні техніко-економічні показники (характеристики) розробленого захищеного сховища даних із використанням технологій блокчейн, визначені у технічному завданні, виконані.

ВИСНОВКИ

У ході аналізу предметної області описано проблеми з конфіденційністю та безпекою зберігання файлів. Рекомендовано впровадження безпечного зберігання файлів, шляхом застосування шифрування для зменшення ризику несанкціонованого доступу до особистих та корпоративних даних. Використання технології блокчейн може забезпечити контроль та незмінність даних. Зазначено, що існуючі сховища для зберігання даних мають свої переваги та недоліки, але найефективнішого рішення, яке поєднує безпеку шифрування та технологію блокчейн, поки немає. Висловлено необхідність використання каскадного шифрування для розв'язання проблеми коливань в ефективності алгоритмів у майбутньому.

Запропоновано модель зберігання даних, де каскадне шифрування використовується для захисту конфіденційної інформації, а технологія блокчейн запобігає підміні даних у файлах. Для розробки було обрано строго типізовану, об'єктно-орієнтовану мову програмування C# та платформу dotnet з відкритим вихідним кодом для розробки web API, а також Angular для веб-клієнту.

Описано алгоритм захищеного збереження даних з використанням технологій блокчейн, проведено аналіз архітектурних підходів та системних архітектур. З використанням UML діаграм показано основні класи системи, варіанти використання, а також відображено схему бази даних для контролю доступу користувачів. Для розробки серверної та клієнтської частини, використанню технологій .NET Angular та Visual Studio 2022, створенню бази даних з використанням підходу Code First та міграцій. Покроково розписано процес розгортання рішення на серверах Azure та локально.

Для забезпечення якості додатку та виявлення помилок проведено тестування модулю шифрування за допомогою юніт-тестів, а також тестування основного функціоналу та інтерфейсу клієнтської частини. Отримані результати додані у відповідну таблицю в додатку.

В економічній частині проведено розрахунки щодо визначення вартості розробки та потенційного доходу. Економічний аналіз показав, що термін окупності інвестицій складає 2,81 років.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Блокчейн: вебсайт. URL: https://uk.wikipedia.org/wiki/Блокчейн (Дата звернення 10.10.2023).
2. How to Store Data in Blockchain: вебсайт. URL: https://www.geeksforgeeks.org/how-to-store-data-in-blockchain/ (Дата звернення 10.10.2023).
3. Waly, M. Learning Microsoft Azure Storage. Birmingham, UK: Packt Publishing. 2017. 368 p.
4. Що таке хмарні технології і навіщо вони потрібні: вебсайт. URL: https://edin.ua/shho-take-xmarni-texnologi%D1%97-i-navishho-voni-potribni/ (Дата звернення 10.10.2023).
5. Ahmad Osama, Shashikant Shakya. Professional Azure SQL Managed Database Administration: Efficiently manage and modernize data in the cloud using Azure SQL, 3rd Edition. Birmingham: Paperback, 2021, 724 p.
6. Mark J. Price, C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code, 6th Edition 6th ed., Packt Publishing Ltd 2021, 824 p.
7. Сховище даних: вебсайт. URL: https://uk.wikipedia.org/wiki/%D0%A1%D1%85%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85 (Дата звернення 10.11.2023).
8. Конфіденційна інформація: вебсайт. URL: https://wiki.legalaid.gov.ua/index.php/%D0%9A%D0%BE%D0%BD%D1%84%D1%96%D0%B4%D0%B5%D0%BD%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D1%8F (Дата звернення 10.11.2023).
9. Siraj Raval. Decentralized Applications: Harnessing Bitcoin's Blockchain Technology 1st Edition. O’Reilly Media•, Inc 2016, 824 p.

10. Understanding ChaCha20 Encryption: A Secure and Fast Algorithm for Data Protection 2023: вебсайт. URL: https://cyberw1ng.medium.com/understanding-chacha20-encryption-a-secure-and-fast-algorithm-for-data-protection-2023-a80c208c1401(Дата звернення 16.11.2023)
11. What Is the Advanced Encryption Standard (AES): вебсайт. URL: https://www.usnews.com/360-reviews/privacy/what-is-advanced-encryption-standard (Дата звернення 16.11.2023)
12. Jon P. Smith. Entity Framework Core in Action First Edition, Manning 2018, 520 p.
13. Latest Block Chain MCQ Objective Question: вебсайт. URL: https://testbook.com/objective-questions/mcq-on-block-chain--5eea6a0d39140f30f369e30d(Дата звернення 18.11.2023)
14. Google drive review: вебсайт. URL: https://www.cloudwards.net/review/google-drive/(Дата звернення 18.11.2023)
15. Amazon S3 review: вебсайт. URL: https://www.cloudwards.net/amazon-s3-review/(Дата звернення 26.11.2023)
16. Dropbox review: вебсайт. URL: https://www.cloudwards.net/review/dropbox/(Дата звернення 26.11.2023)
17. Microsoft Azure review: вебсайт. URL: https://www.cloudwards.net/microsoft-azure-review/(Дата звернення 16.11.2023)
18. Data Storage Security: вебсайт. URL: https://hypertecsp.com/knowledge-base/data-storage-security/(Дата звернення 16.11.2023)
19. Data Storage Security: Best Practices for High Cyber-Resilience: вебсайт. URL: https://www.infopulse.com/blog/data-storage-security-best-practices(Дата звернення 16.11.2023)
20. В. М. Богуш, В. В. Богуш, В. Д. Бровко, В. П. Настрадін; під. ред. В. М. Богуша. Основи кіберпростору, кібербезпеки та кіберзахисту. Навч. посіб. — К.: Видавництво Ліра-К. 2020. — 554 с.
21. Rubin F. Secret Key Cryptography: Ciphers, from Simple to Unbreakable. Shelter: Manning, 2022.−344p.

22. Hash function Second Edition Paperback – November 21, 2021 by Gerardus Blokdyk (Author) 306 p.
23. Sam Newman. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith 1st Edition, O'Reilly - 2019, 284 p.
24. Microservices-vs-monolith: вебсайт. URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith(Дата звернення 28.11.2023)
25. Melanie Swan. Blockchain: Blueprint for a New Economy 1st Edition. North Sebastopol: O'Reilly Media, 2015, 149 p.
26. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти: вебсайт. URL: https://dou.ua/forums/topic/40575/(Дата звернення 28.11.2023)
27. Діаграма компонентів: вебсайт. URL: https://uk.wikipedia.org/wiki/Діаграма_компонентів(Дата звернення 28.11.2023)
28. Joshua S. Ponelat and Lukas L. Rosenstock. Manning publishing, May 2022, 424 p.
29. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. / Укладачі В.О. Козловський, О.Й. Лесько, В.В.Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
30. CLI Overview and Command Reference: вебсайт. URL: https://angular.io/cli(Дата звернення 29.11.2023)
31. Catherine Newbould. Software Testing Foundation Level Guide for ISTQB certification Syllabus v4. AiFlex Publishing, November 21, 2023, 382 pages.

ДОДАТКИ

98

Додаток А (обов'язковий)
Технічне завдання

ЗАТВЕРДЖУЮ
Завідувач кафедри АПТ
д.т.н., проф. Олег БІСІКАЛО
«12» жовтня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

Розробка захищеного сховища даних із використанням технологій блокчейн»

08-31.МКР.011.02.000 ТЗ

Керівник роботи
к.т.н.,доц.каф.АПТ. Ярослав КУЛИК
«12» жовтня 2023 р.

Виконавець:
ст. гр. 1ІСТ-22м
Степан СІДАК
«12» жовтня 2023 р.

Вінниця ВНТУ 2023

1. Назва та галузь застосування

Магістерська кваліфікаційна робота: «Розробка захищеного сховища даних із використанням технологій блокчейн». Галузь застосування – інформаційні технології.

2. Підстава для розробки

Розробку системи здійснювати на підставі наказу по університету № __від__.__ 2023 та завдання до магістерської кваліфікаційної роботи, складеного та затвердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій».

3. Мета та призначення розробки

Метою роботи є зменшення ймовірності компрометації файлів та втрати приватних даних.

4. Джерела розробки

1. В. М. Богуш, В. В. Богуш, В. Д. Бровко, В. П. Настрадін; під. ред. В. М. Богуша. Основи кіберпростору, кібербезпеки та кіберзахисту. Навч. посіб. — К.: Видавництво Ліра-К. 2020. — 554 с.

2. Data Storage Security: Best Practices for High Cyber-Resilience: вебсайт. URL: https://www.infopulse.com/blog/data-storage-security-best-practices(Дата звернення 16.11.2023).

3. Siraj Raval. Decentralized Applications: Harnessing Bitcoin's Blockchain Technology 1st Edition. O’Reilly Media•, Inc 2016, 824 p.

5. Показники призначення

Основні технічні вимоги та мінімальні системні вимоги до програми:

- ОС: MacOS, Windows, Linux.
- Двоядерний процесор.
- 4 гігабайти оперативної пам’яті.
- Час роботи шифрування – до 20 секунд на 1 гігабайт даних

Результати роботи програми:

- Завантаження файлу;
- Перегляд завантажених файлів;
- Наявність системи авторизації;
- Шифрування та розшифрування файлів;

6. Економічні показники

- витрати на розробку – не більше 100 тис. грн;
- абсолютний ефект від впровадження розробки – не менше 900 тис. грн;
- внутрішня дохідність інвестицій – не менше 42%;
- термін окупності – не більше 3 років.

7. Стадії розробки

1. Розділ 1 «Аналіз предметної області та огляд існуючих систем сховищ даних» має бути виконаний до 19.09.2023.
2. Розділ 2 «Опис системи та методи і моделі збереження даних» має бути виконаний до 10.11.2023.
3. Розділ 3 «Аналіз програмної реалізації захищеного сховища даних» має бути виконаний до 16.11.2023.
4. Розділ 4 «Програмна реалізація захищеного сховища даних із використанням технологій блокчейн» має бути виконаний до 16.11.2023.
5. Економічний розділ має бути виконаний до 19.11.2023.

8. Порядок контролю та приймання

1. Рубіжний контроль провести до 07.12.2023.

2. Попередній захист магістерської кваліфікаційної роботи провести до 21.11.2023.

3. Захист магістерської кваліфікаційної роботи провести до 19.12.2023.

Додаток Б (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

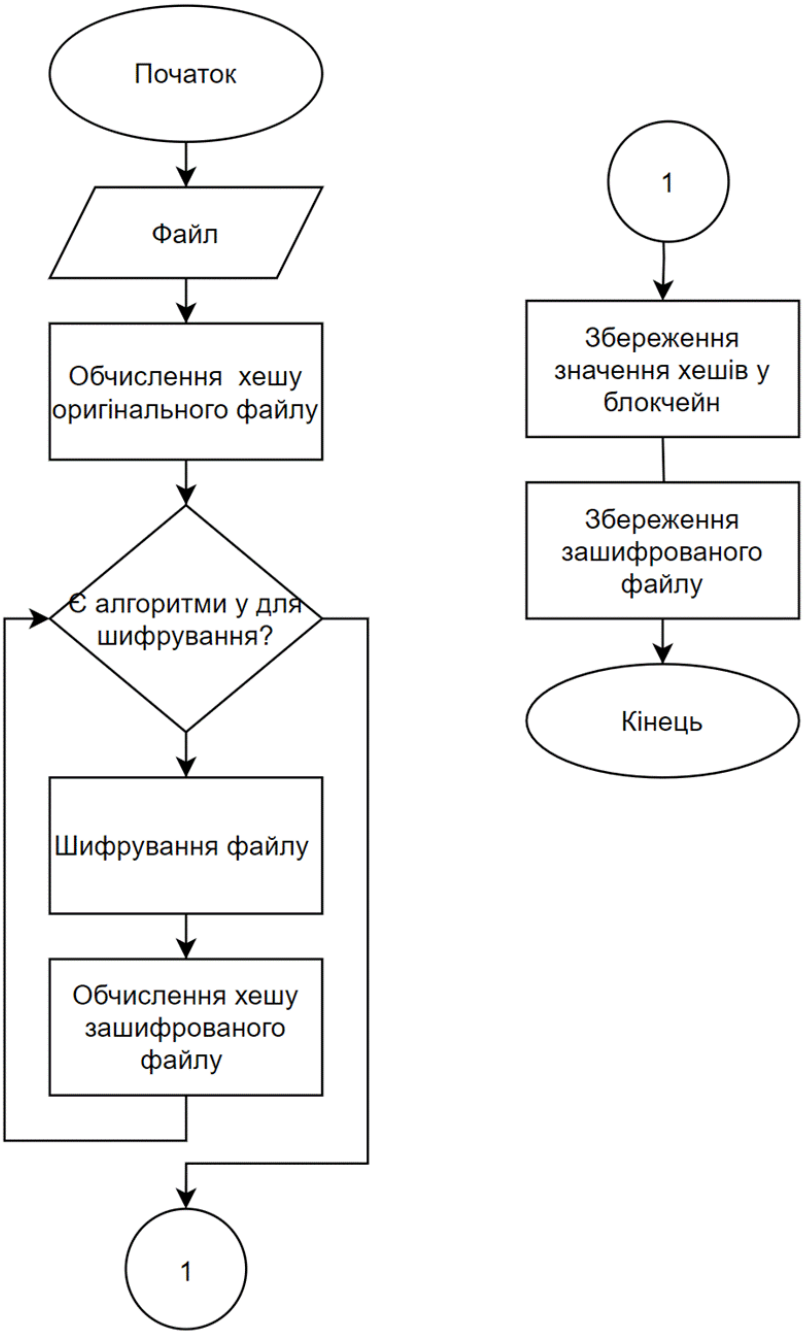

Рисунок Б.1 – Блок схема алгоритму шифрування файлів із використанням технології блокчейн та каскадного шифрування

Продовження додатку Б:

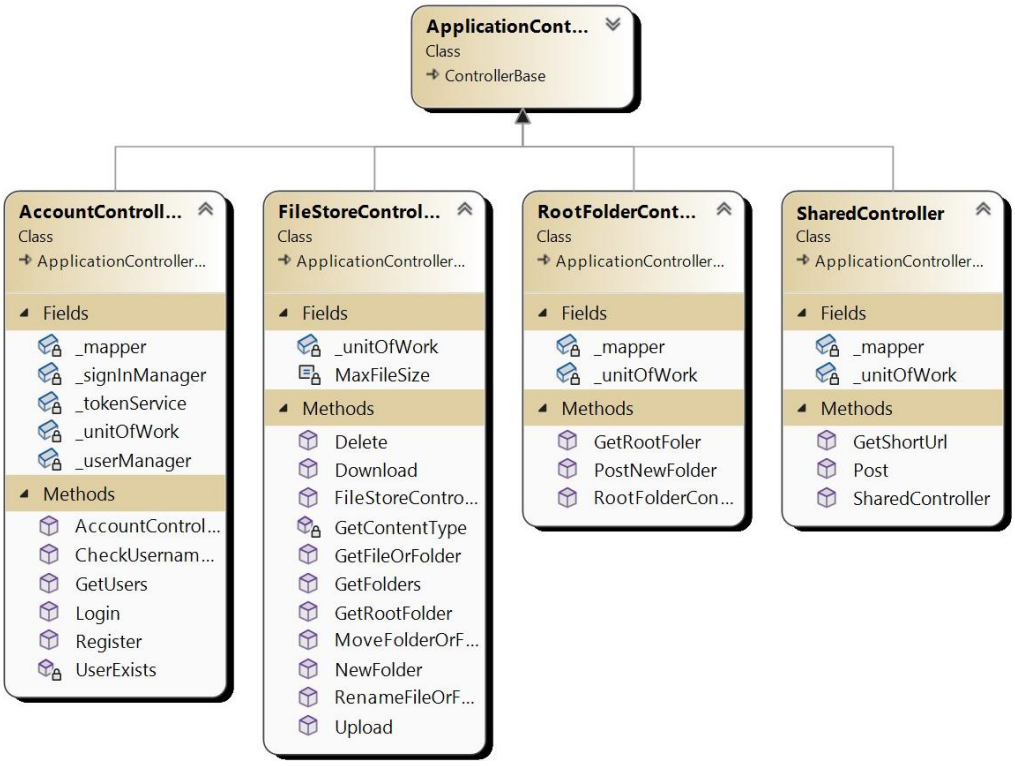

Рисунок Б.2 – Діаграма класів веб серверу

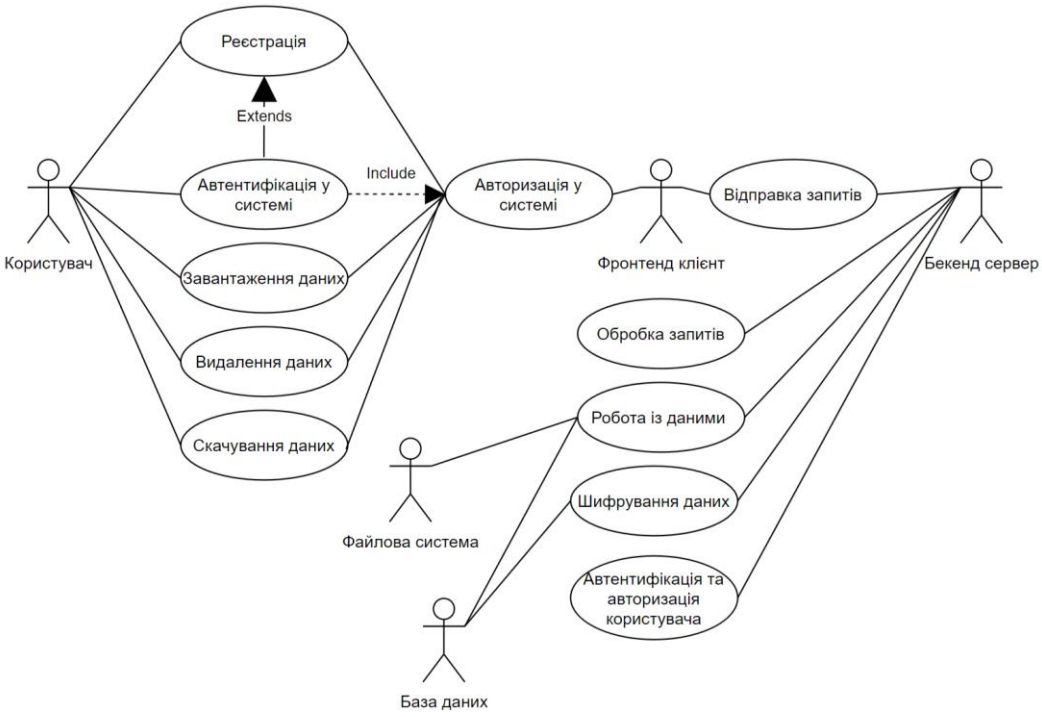

Рисунок Б.3 – Діаграма варіантів використання

Продовження додатку Б

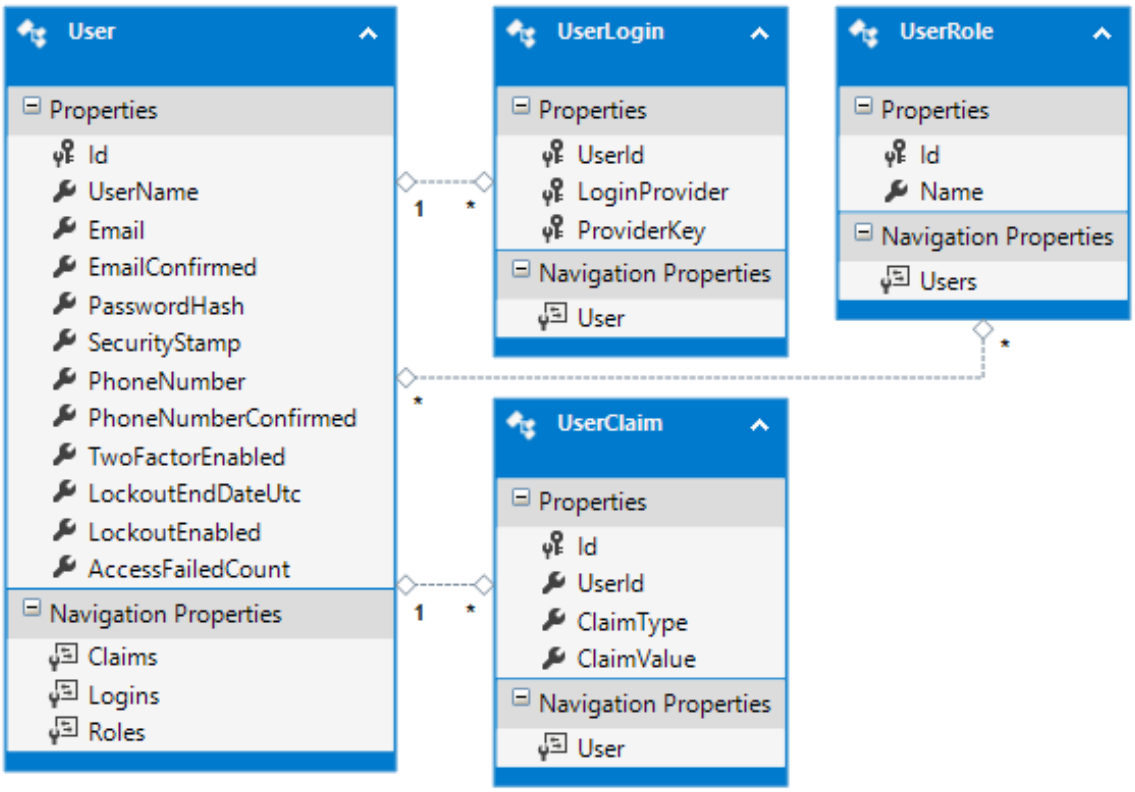

Рисунок Б.4 – Схема таблиць бази даних

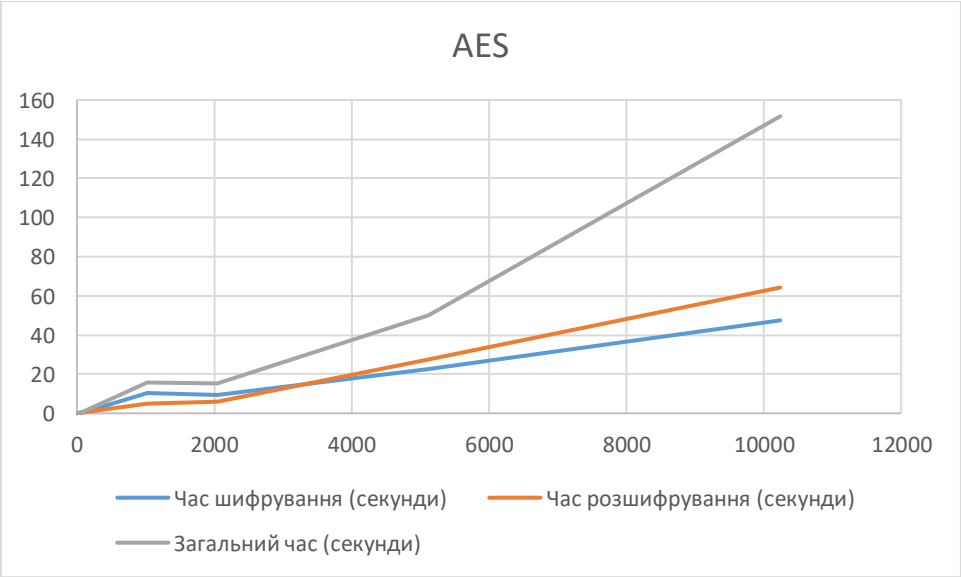

Рисунок Б.5 – Графік часу для операцій із використанням алгоритму AES

Продовження додатку Б

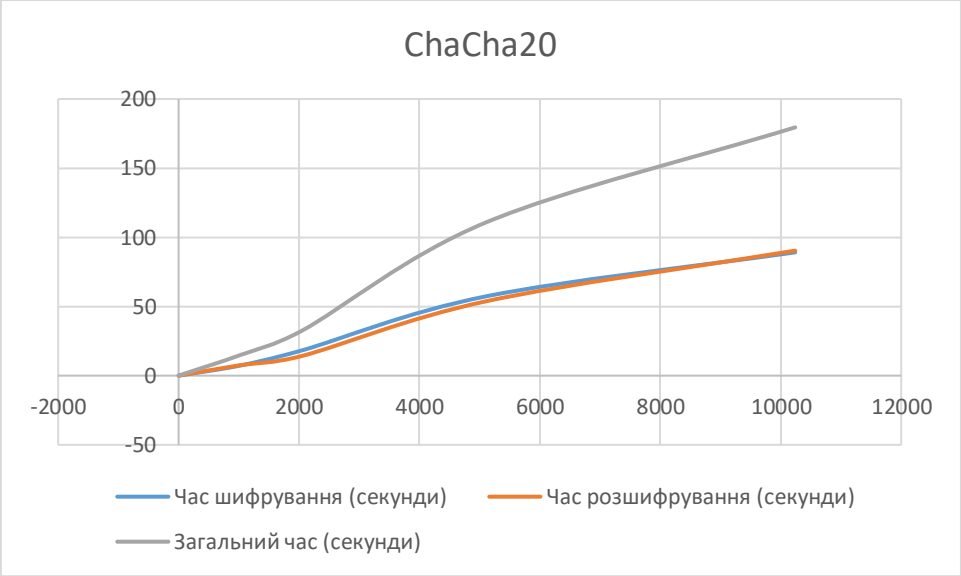

Рисунок Б.6 – Графік часу для операцій із використанням алгоритму ChaCha20

Додаток В (обов'язковий)
Тестування додатку

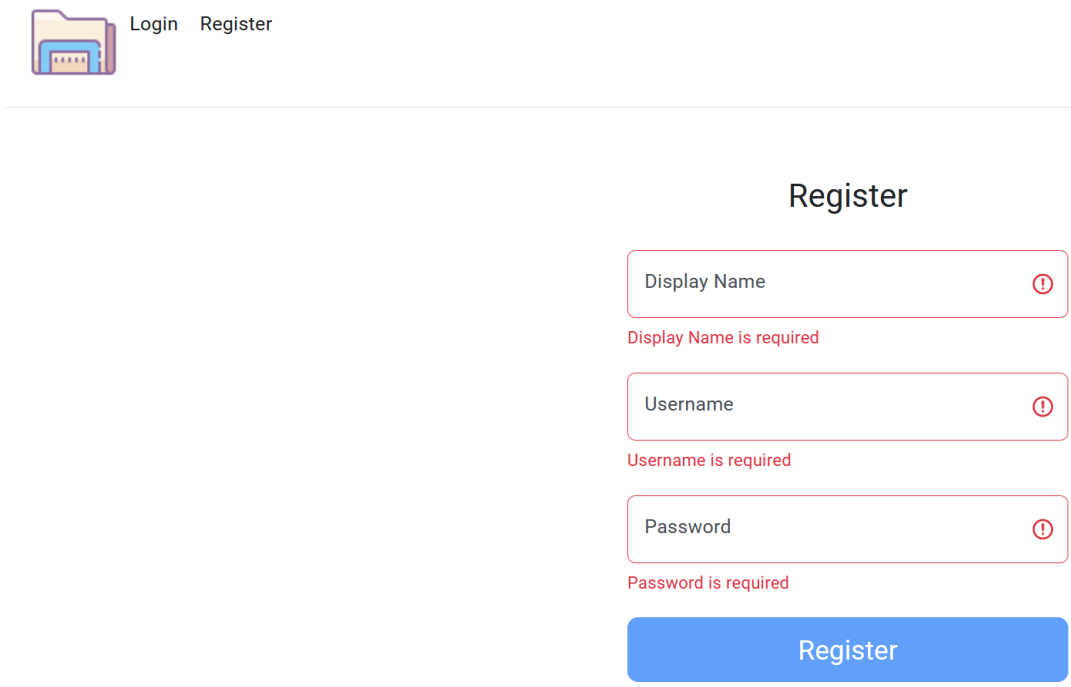

Рисунок В.1 – Тестування полів введення форми реєстрації

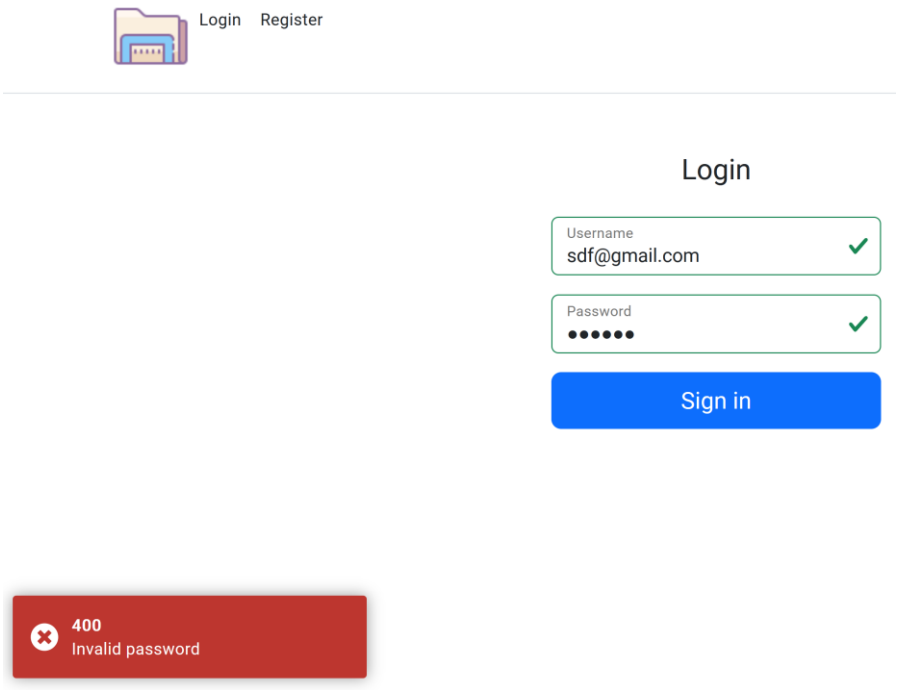

Рисунок В.2 – Помилка при спробі входу в систему із невірними даними

Продовження додатку В

Таблиця В.1 – Час для операцій із використанням алгоритму AES

Розмір файлу (мегабайти)	Час шифрування (секунди)	Час розшифрування (секунди)	Загальний час (секунди)
1	0.06042	0.01406	0.07448
10	0.04884	0.04169	0.09053
100	0.44232	0.51266	0.95499
1024	10.41966	5.10193	15.52159
2048	9.41412	6.04424	15.45835
5120	22.58613	27.30243	49.88856
10240	47.45883	64.22546	151.68429

Таблиця В.2 – Час для операцій із використанням алгоритму ChaCha20

Розмір файлу (мегабайти)	Час шифрування (секунди)	Час розшифрування (секунди)	Загальний час (секунди)
1	0.0831	0.0163	0.0994
10	0.0816	0.0744	0.1561
100	0.7315	0.7513	1.4828
1024	7.3406	7.5767	14.9173
2048	18.265	14.2365	32.5016
5120	57.4653	53.8984	111.05
10240	89.1451	90.365	179.50

108

Додаток Г (обов'язковий)
Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка захищеного сховища даних із використанням технологій блокчейн

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ: кафедра Автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 93.8% Схожість 6.2%

Аналіз звіту подібності (відмітити потрібне):

☑ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Роман МАСЛІЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Степан СІДАК
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Ярослав КУЛИК
(підпис) (ім'я, ПРІЗВИЩЕ)